Indice

Resumen:

Para conseguir la personalización de la enseñanza asistida por computador o e-learning es necesario la división de los cursos en Objetos Educativos. Dichos Objetos Educativos deberán estar lo suficientemente documentados con metadatos para poder almacenarlos y gestionarlos en repositorios. Dichos metadatos deberán seguir unos estándares para que pueda ser entendido por cualquier sistema. En este proyecto se propone un estudio de dichos esquemas y una herramienta, MetaRDF. MetaRDF es una herramienta para la creación y edición de metadatos. Los esquemas utilizados en dicha herramienta son DC (Dublín Core), LOM (Learning Object Model) y LOMe (esquema basado en LOM propuesto en este proyecto para e-Aula). La clasificación descriptiva ofrecida por los estándares de metadatos se cumplimenta mediante taxonomías, que aportan significado a los contenidos, iniciando el camino hacia la web semántica.

Resumen en ingles:

To achieve the e-learning personalization the courses are required to be divide into Learning Objects. These Learning Objects have to be documented enough with metadata so it is possible to store and manage them in repositories. This metadata will have to follow some standards so any system can understand them. In this proyect, a study of these standards and a tool, Meta RDF, are proposed. MetaRDF is a tool for metadata creation and edition. The standard used in this tool are DC (Dublin Core), LOM (Learning Object Model) and LOMe (and adaptation for e-Aula based in LOM). The descriptive classification offered by the metadata standards are cumplimented by the taxonomies, which give meaning to the content, starting the way that goes to the semantic Web.

Palabras clave:

Enseñanza asistida por ordenador, E.A.O, e-learning, metadatos, metainformación, RDF, estándares educativos, LOM, DC, LOMe.

1 - Contexto

1.1 - ¿Qué es e-learning?

"Es el conjunto de actividades necesarias para la creación y uso de un entorno de formación a distancia on-line mediante el uso de tecnologías de la información y comunicaciones" (Fuente: http://www.campusformacion.com/glosario.asp)

Es decir, el e-learning es la educación basada en las nuevas tecnologías. El desarrollo de éstas ha abierto un campo de infinitas posibilidades a la enseñanza. La informática y concretamente Internet ha supuesto la demolición de gran cantidad de barreras con las que se encontraba la enseñanza tal y como se entiende en las aulas.

La explosión de Internet y las nuevas tecnologías permiten el acceso a la formación en cualquier momento y lugar, sacando la educación del entorno tradicional de las aulas y permitiendo que los alumnos sigan su propio ritmo de trabajo. Además posibilitan el intercambio información entre los diferentes agentes que participan en el proceso educativo (profesor-profesor, profesor-alumno y alumno-alumno) y la colaboración y el trabajo en grupo,

Uno de los objetivos de las nuevas tecnologías e-learning es llegar a conseguir cursos generados dinámicamente en función de las necesidades específicas de cada alumno, sus particularidades a la hora de adquirir conocimiento (estilos de aprendizaje) y sus preferencias. De esta manera, se optimizaría considerablemente el tiempo que los alumnos emplean en su formación. En una sociedad donde cada vez hay más demanda de profesionales con conocimientos permanentemente actualizados y en la que se dispone de poco tiempo para dedicar a la formación, puesto que se tiene que compaginar con la actividad profesional, la demanda de aplicaciones que permitan personalizar la formación en función de las características específicas es cada vez mayor.
Con la nueva generación de aplicaciones e-learning, cada alumno puede adaptar la velocidad a la que recibe el curso en función de sus capacidades, ya que él podrá decidir si repetir una lección o pasarla más rápidamente si la ha comprendido sin dificultad. Por otro lado, se puede conseguir el objetivo que busca la sociedad actualmente, aprender y seguir aprendiendo siempre, la educación permanente (*long-life learning*) es posible gracias a este nuevo concepto de enseñanza. Ya no es necesario asociar la etapa de aprendizaje con la juventud y suponer que la madurez ya no implica seguir aprendiendo. Además desaparecen las barreras que encierran la educación en un momento y lugar concreto, en el contexto de un aula, puesto que es posible hacer llegar al alumno contenidos adaptados a sus necesidades concretas en el momento en que le son necesarios.

Con la enseñanza tradicional, hacer llegar los conocimientos de una única persona a los demás puede resultar un proceso largo y tedioso con una gran cantidad de esfuerzo por parte de la persona que desea compartir su sabiduría. Con el e-learning se hace posible distribuir la información a lo largo y ancho sin grandes costes. Es más, facilita el intercambio de dichos conocimientos para permitir el enriquecimiento de otros cursos relacionados.

El uso de ordenadores para la enseñanza se empezó a utilizar en la década de los 60, pero en sus comienzos aún tenía muchas limitaciones. Con la explosión de Internet se potenció enormemente tanto el intercambio de información como la posibilidad de que la misma llegue a cualquier punto conectado a la Red. Ha sido a partir de Internet cuando el e-learning ha sufrido una gran revolución superando una gran barrera y permitiendo su amplia difusión.
Por lo tanto el e-learning ha supuesto toda una revolución con respecto a la idea tradicional que tenemos de enseñanza. Gran cantidad de organismos han adoptado esta nuevas tendencias obteniendo muy buenos resultados.

Algunos ejemplos de plataformas de e-learning son:

- Campusformación (www.campusformacion.com). Nos ofrece una interfaz sencilla e intuitiva con una gran potencia para crear cursos y manejar gran número de usuarios. Actividades diversas para cada curso creado por los profesores: chats, foros, tests, mensajería interna, tareas, actividades en grupo... Registro y seguimiento completo de los accesos del alumno. Los alumnos pueden subir sus tareas al servidor. Dispone de un completo sistema de evaluación y seguimiento, informes de actividad de cada estudiante, con gráficos y detalles sobre su paso por cada módulo. Permite mostrar cualquier contenido digital, Word, PDF, PowerPoint, Flash, vídeo, sonidos, etc.
- WebCT (http://www.webct.com). Es una herramienta software con la intención de proporcionar un medio para el desarrollo de material docente a impartir, mediante el uso de tecnologías web.
  La idea de usar WebCT es la de crear un conjunto de “aulas virtuales” en las que cada profesor y alumno tengan su propia área de desarrollo.
  Los profesores podrían publicar los contenidos del curso, realizar evaluaciones a los alumnos, controlar el acceso, y a cada parte concreta del curso, asignar tareas, comunicarse con los alumnos de forma individual y colectiva, así como muchas otras funciones de control y evaluación.
  Los alumnos podrían seguir los cursos, realizar los trabajos encomendados y auto evaluaciones para conocer el progreso en la asignatura, participar en foros electrónicos sobre su contenido, publicar mensajes en tablones de anuncios, contactar con el profesor a través de correo electrónico, etc... todo ello supervisado por un profesor. Instalada en la Universidad de Cantabria.
- Adamadrid (http://adamadrid.uc3m.es/). Plataforma de la Universidad Complutense de Madrid que ofrece enseñanza a través de Internet con foros, chats, biblioteca con enlaces a documentos clasificados por temas, sección de avisos para los alumnos... El alumno dispone de los temas on-line siempre que quiera. Evaluación mediante tests de cada tema, con control del tiempo empleado en cada prueba mediante la sesión abierta por el alumno. Todo esto complementado con sesiones de videoconferencia con el profesor.

1.2 - Qué ha supuesto la explosión de la Web para aplicaciones de la enseñanza asistida por computador (Computer Assisted Instruction CAI)

La enseñaza asistida por computador (Computer Assisted Instruction CAI) nació en los años 60 en los EEUU, basándose en la Enseñanza Programada desarrollada por el psicólogo Skinner a finales de los años 50. Este método de enseñanza inicial era

completamente lineal, en el que se iba siguiendo siempre un mismo camino a través del cual se iban adquiriendo los conocimientos pertinentes. En la misma época nace otro método de enseñanza no lineal (Crowder), en la que no se sigue siempre un esquema concreto, sino que se tiene la posibilidad de variarlos según las circunstancias.

Pero lo que finalmente ha supuesto una revolución para este tipo de enseñanza ha sido el desarrollo de las nuevas tecnologías, el hecho de que ahora en casi todas las casas haya un pc y que Internet se ha extendido enormemente ha supuesto un cambio muy importante en la enseñanza por ordenador o e-learning.
El desarrollo de las nuevas tecnologías permite la creación de programas educativos muy complejos en los que, más allá de un texto con imágenes, se obtiene pleno aprovechamiento de las posibilidades multimedia: secuencias animadas de imágenes y video, sonidos y música, búsquedas y enlaces dinámicos, configuración según las preferencias del usuario, etc.

Definitivamente Internet ha supuesto un cambio radical en la idea tradicional que siempre hemos tenido de la enseñanza. La Red se ha convertido en un complemento indispensable para la enseñanza por computador, transformándola en algo mucho más accesible y cercano.

Internet ha dado mayor flexibilidad al e-learning. De forma que el estudiante puede recibir formación en el momento y lugar que desee con la única condición de tener conexión, sin tener que estar acordándose de llevar su curso en el soporte que sea de un lado para otro. Así, un estudiante puede adaptar su propia línea de aprendizaje a su gusto. Resulta mucho más sencilla la gestión y administración de los datos llegando al alumno de forma mucho más rápida y eficaz.

Este punto está íntimamente relacionado con otra de las ventajas que ha supuesto Internet para el aprendizaje por computador, hablamos de la fácil difusión de la información. Gracias a la Red, la enseñanza no tiene límites y puede llegar fácilmente a todo el que disponga de un pc con conexión. Esto supone un gran adelanto en lo que llamamos long-life learning ya que todo el mundo, tiene acceso fácil a nuevos conocimientos, por lo que el proceso de aprendizaje ya no se limita únicamente a una etapa de nuestra vida. Sino que está distribuida durante tanto tiempo como queramos.

Internet permite el intercambio de contenidos entre personas a lo largo de todo el planeta, independientemente de su localización física. Esto permite a los creadores de los cursos, tener acceso a otros repositorios de manera muy sencilla, de forma que pueden complementar sin grandes esfuerzos los temas que ellos crean.

Así, se garantiza al alumno que los contenidos que está aprendiendo son los de última generación, gracias a la capacidad de adaptación y renovación constante de contenidos y datos sin esfuerzo. Ya que si el creador del curso tiene acceso a múltiples repositorios desde su ordenador, le resultará muy sencillo actualizar sus conocimientos y por tanto los que transmite. De igual manera, cada profesor puede poner a disposición de los demás todo su saber.

El intercambio que facilita Internet, no es bueno solamente para que los profesores completen sus cursos y ofrezcan los últimos avances en el tema a tratar. Además, gracias a los foros, chats... etc. Se permite al profesor ponerse en contacto con el alumno sin la necesidad de estar físicamente juntos, con la ventaja que esto supone al poder atender dudas de varios alumnos sin gastos de tiempo en desplazamientos. Es más, permite el intercambio de información entre alumnos, de

forma que lo que uno descubre por su cuenta puede compartirlo con sus compañeros de curso que pueden estar buscando lo mismo sin haber tenido éxito.

La reducción de costes también es un punto importante. Ya no hace falta que cada alumno tenga una copia física del curso a seguir para realizarlo. Con que haya una única copia y se pueda acceder a través de Internet es suficiente, con el ahorro de material que conlleva. Y no solo eso sino además permite al alumno acceder en sitios distintos dependiendo de donde se encuentre sin la necesidad de estar "cargando" con el soporte físico del curso donde quiera que vaya.

1.3 - Búsqueda de dos objetivos: reutilización y adaptación de los contenidos.

Un efecto colateral de la extraordinaria expansión de la Web y de las aplicaciones de educativas e-learning es la enorme cantidad de contenidos educativos que se han creado para este fin. El hecho es que en estos momentos existen una enorme cantidad de contenidos educativos de una gran calidad, por lo que la idea de poder reutilizar este material para cursos diferentes aparece es de un enorme atractivo para los creadores de cursos, puesto que significaría, menor esfuerzo de creación, menores costes y mejorar la calidad de los cursos.
Sin embargo, reutilizar un curso completo es difícil, puesto que está concebido para un contexto y un uso concretos y difícilmente se puede emplear en otro contexto diferente. Por esta razón, hace algunos años, surgió la idea de aplicar el paradigma de la programación orientada a objetos a la creación de cursos. Cursos compuestos de piezas autocontenidas, recombinables y reutilizables. A estas piezas educativas en las que se disgregan los cursos, que son susceptibles de ser reutilizadas en diferentes contextos, se les llamó objetos educativos u objetos didácticos (learning objects, en inglés).
De manera intuitiva, las características que deben reunir dichos componentes reutilizables son la independencia semántica frente a otros objetos y un tamaño lo suficientemente pequeño como para poder incluirse con facilidad en otros contextos.

Evidentemente, la reutilización está ligada a la interoperabilidad. Poder reutilizar contenido pasa porque esos contenidos puedan utilizarse en distintas plataformas y sistemas.
De acuerdo con el IEEE la interoperabilidad se define como "la habilidad de dos o más sistemas o componentes para intercambiar información y para usar la información que ha sido intercambiada". En los sistemas de e-learning, la interoperabilidad permite el intercambio y reutilización de recursos educativos (cursos, documentos, videos, tutoriales, etc…) que han sido desarrollados en plataformas educativas heterogéneas, lo cual permite:

- Incrementar la calidad y variedad de recursos disponibles en el mercado.

- Preservar el capital invertido en tecnología y desarrollo de recursos educativos, ya que un recurso educativo podrá ser intercambiado o usado sin la necesidad de realizar costosas modificaciones.

- Garantizar que los usuarios con diferentes plataformas HW y SW puedan acceder a recursos educativos de fuentes heterogéneas, con perdidas mínimas de contenido y funcionalidad.

La idea de cursos compuestos por piezas reutilizables implica también un incremento de la capacidad de adaptación en función de las características específicas de los alumnos.
Para que sea posible reutilizar objetos y adaptar cursos combinando los objetos adecuados en función del usuario, no sólo es imprescindible la interoperabilidad, sino que es necesario también recuperar dichos objetos, que en teoría pueden encontrarse distribuidos en cualquier repositorio de la red. Por tanto es imprescindible disponer de información que permita la localización de dichos objetos de acuerdo a criterios específicos.

1.4 – Modelo de objetos educativos

La baza fundamental del e-learning es la personalización. Ya que cada alumno seguirá el curso por su cuenta y a su ritmo ¿porqué no adaptar cada curso a sus necesidades? Podríamos basar la creación del curso en función de los conocimientos previos del alumno y de los objetivos que pretende alcanzar con el mismo. Esto se conseguiría dividiendo el curso en pequeños módulos que pudiésemos recombinar en función de la persona que quiera tomarlo. Pero estos módulos deben de ser lo suficientemente pequeños como para que se puedan mezclar con facilidad, es más, deben de estar lo suficientemente documentados como para poder unirlos a otros fácilmente. Ya que sabríamos sin dificultad qué objetivos se consiguen con ese módulo y qué prerrequisitos tiene. Por lo tanto esos módulos no serán temas ni unidades, serán cosas más pequeñas como un gráfico, un video, un ejercicio... etc.

Pero esto tiene más implicaciones, la reutilización y la localización.

Al tener tantos módulos fácilmente combinables se puede pasar a la reutilización y compartición de los mismos. Un mismo módulo puede servir para múltiples cursos y para varias personas. Por lo tanto es buena idea crear repositorios donde esos módulos estarán almacenados y donde cualquiera que quiera crear un curso pueda entrar para coger los que más le convengan, aumentando así su valor con cada reutilización.

Pero esta reutilización implica una organización exhaustiva y que la localización de dichos módulos sea sencilla. Que sea fácil localizar justo el que se necesita en un momento determinado, o que no haga falta consultar el módulo en si para saber si es lo que queremos se puede conseguir asociándole una información sobre sí mismo, una meta información. Pero para que éstos metadatos puedan ser consultados fácilmente de una forma sistemática tanto por una persona como por una herramienta hace falta que todos sean construidos de la misma manera, que todos se compongan de los mismos puntos, es decir, que sigan un estándar.

Estos módulos en los que se dividirán los cursos es lo que llamaremos objetos educativos, que según el IEEE son:

"Any entity, digital or non-digital, that may be used for learning, education or training." -- IEEE 1484.12.1-2002, 15 July 2002, Draft Standard for Learning Object Metadata, IEEE Learning Technology Standards Committee (LTSC)

Pero esta definición se nos queda corta. Esta definición es muy general, y no entra concretamente en las características que debe tener un OE (Objeto Educativo). Encontramos otras definiciones igualmente débiles como:

"any digital resource that can be reused to support learning" -- David A. Wiley, "Connecting Learning Objects to Instructional Design Theory"

Ambas definiciones se nos quedan cortas. Veamos concretamente qué atributos debe tener un OE:

- Son la unidad independiente de información más pequeña que puede proporcionar un conocimiento concreto. Por lo tanto debe ser un elemento breve más pequeño incluso de lo que solemos considerar un tema de una asignatura.

- Son almacenados y recuperados a través de sus metadatos.

- Se unen para formar un curso en función de la información aportada por sus metadatos. Dichos metadatos deben incluir el contenido formativo, los requisitos, la forma de presentación y los objetivos.

- Sus metadatos deben de seguir un estándar para uniformizar la creación y recuperación de dichos Objetos Educativos

- No debe hacer referencia a ninguno otro Objeto Educativo ni debe de depender de la presencia de otro para formar parte de un curso.

Ventajas de los Objetos Educativos:

- Incremento del valor del contenido. Cada vez que se reutiliza el contenido del OE se incrementa su valor, ya que cada vez que se reutiliza está suponiendo un ahorro para el que lo maneja ya que no ha tenido que crearlo. Se genera una vez y se utiliza múltiples veces en distintos contextos.

- El ser contenidos tan independientes no será necesario rescribirlos según el contexto cada vez que se utilice, con esto queremos señalar su flexibilidad.

- Mejora en la actualización, búsqueda y gestión del conocimiento en general. Al estar los OE perfectamente documentados con los metadatos, todas estas acciones se simplifican. La mayor ventaja de los OE es que gracias a ellos se administran los contenidos de los cursos con mucha facilidad y precisión.

- Permiten que los cursos sean más personalizados ya que éstos se podrán hacer a medida para cada alumno según su perfil.

1.5 - ¿Qué son los metadatos? Necesidad de metadatos

Los metadatos son información sobre los datos. Son información asociada a determinados datos con la finalidad de clasificarlos, de saber cuál es su contenido, sus características sin acceder a él. Para hacernos una idea los metadatos serían la información que se asocia a cada libro en una biblioteca, el título, el autor, la editorial, el tema… es decir, toda esa información que nos permitirá clasificar dicho libro y encontrarlo de forma rápida y eficaz.

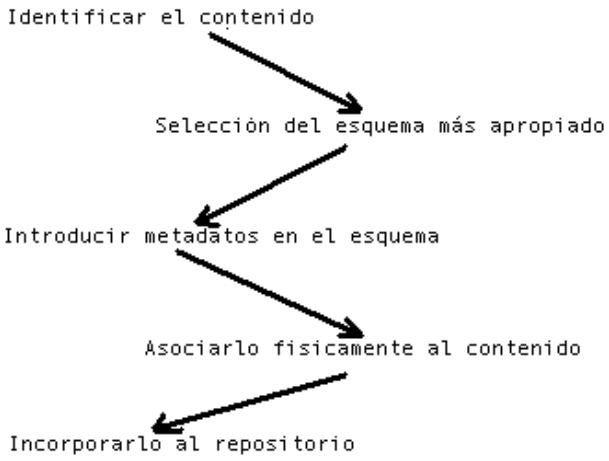
Los metadatos en e-learning son la base para la gestión de los objetos educativos a la hora de construir cursos. Es información sobre la información que permita a los usuarios saber si el OE es útil para ellos y puedan encontrarlo rápidamente.
Los metadatos son lo que necesitamos saber sobre un módulo de un curso para poder ensamblarlo con los demás. Se trata de hacer clasificaciones estructuradas de los OE para catalogar la información de acuerdo a determinados criterios. Información como el lenguaje en el que está escrito, la fecha de creación, el formato que tiene, el tiempo que se tardaría en asimilar los conocimientos que ofrece ese módulo... Por lo tanto los metadatos en e-learning van más allá. No solo nos dan información para que sea más fácil recuperar el contenido sino que también nos habla sobre la relación que tiene con otros contenidos, como se comporta ese contenido, su función, cómo se debe manejar y su uso.

En el caso del e-learning los metadatos serán no solo la información en si, sino también la estructura con la que los representamos y las categorías que comprenden esta estructura.

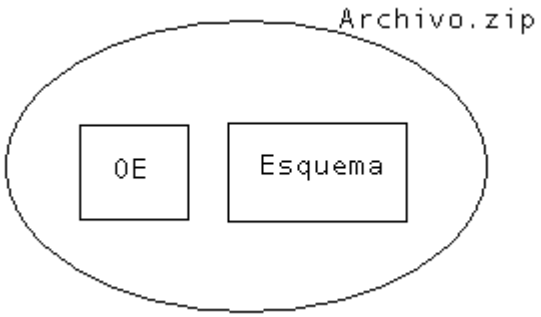
Los metadatos deben de seguir un estándar, aquí es donde entran los lenguajes de marcado (RDF en nuestro caso) que nos ayudarán a estructurar esta información siempre de una manera concreta. Los esquemas son muy importantes, el seguir un estándar permite garantizar el intercambio de información con éxito.
Almacenaremos la metainformación entre etiquetas de dicho lenguaje de forma que sea fácilmente recuperable por una herramienta de creación de cursos. Con los lenguajes de marcado conseguiremos que los metadatos se almacenen en forma de esquemas de forma que cada punto pueda ser recuperado independientemente según la información que nos convenga recuperar.

Actualmente hay en circulación varios esquemas que deberán elegirse según lo que queramos clasificar. En el caso de los recursos educativos, se trata de aportar información sobre cosas que tienen utilidad (o deberían tenerla) desde el punto de vista de la educación. Dichos esquemas van desde la simplicidad del DC (Dublín Core) que tiene solo 15 puntos, es muy genérico y está concebido para recursos en general, hasta la complejidad del LOM (Learning Object Metadata), que consta de tantos puntos que éstos están clasificados en 9 categorías y es el único estándar reconocido.

El proceso de creación de metadatos sería como sigue:



El paso de asociar físicamente los metadatos al OE es importante, porque así nos ahorraremos la creación de una infraestructura de clasificación de metadatos teniendo en un único repositorio el OE y el metadato unido. En la herramienta desarrollada en este proyecto se consigue esta asociación comprimiendo en un zip el OE y el esquema relleno con los metadatos. Ver figura siguiente.



La información que contienen los metadatos podría dividirse en:

- Información estructural. Características del contenido en si que influirán a la hora de combinarlo con otros OE para formar un curso. Características como pueden ser en qué lenguaje está realizado el módulo de forma que el curso final sea homogéneo, o el tiempo que se tardaría en obtener los conocimientos correspondientes a ese módulo.
- Información técnica como qué soporte software se necesita para visualizarlo, versiones que necesita, tamaño del archivo... para que la creación de cursos en distintas plataformas sea posible.
- Información de control de versiones de dichos Objetos Educativos, almacenando la fecha, el nombre del creador... etc

Por lo tanto se hace evidente la necesidad de la metainformación, sin los metadatos la creación de cursos a medida no es posible. Porque ellos serán los criterios a seguir a la hora de crear un curso, serán la base para que una herramienta decida

si ese módulo combina bien con tal otro y en qué orden para la consecución de unos objetivos finales.

A la hora de crear un curso a partir de módulos es imprescindible tener toda la información necesaria de los mismos para que el resultado final sea algo homogéneo, sin lagunas ni grandes saltos en el nivel de conocimientos previos requeridos.

La necesidad de los metadatos se pueden resumir en los siguientes puntos:

- Con las cantidades de información que se tienen hoy en bibliotecas, museos, Internet… se hace imprescindible tener una forma de clasificación. Los metadatos nos ofrecen la forma de tener unos datos básicos de esos contenidos de forma bien esquematizada que nos permitirá clasificarlos y saber en todo momento donde está cada contenido.

- Con los metadatos garantizamos que la información no se perderá nunca. Con la clasificación que hemos descrito antes conseguiremos tener los contenidos de forma que no se ahoguen entre toda esa ingente cantidad de información.

- También nos permiten relacionar unos contenidos con otros. Al realizar la clasificación anteriormente explicada podremos crear vínculos entre unos contenidos y otros siguiendo diferentes criterios.

- Hacer los contenidos más accesibles. Si éstos están bien documentados con sus metadatos correspondientes la recuperación y manejo de contenidos se simplificará mucho a la vez que se hará que las búsquedas sean mucho más precisas.

- Facilita el intercambio de información y su reutilización incrementando así el valor del contenido. Si tenemos nuestros contenidos bien documentados el intercambio con otros usuarios será rápido y eficiente, incrementando el valor de la información con cada intercambio.

1.6 - Necesidad de estándares. LOM. ¿Por qué LOM es insuficiente frente a los objetivos de adaptación y reutilización?

En el momento en que necesitamos que los contenidos educativos sean reutilizables y adaptables para más de un contexto, necesitamos un estándar o normas que fijen el formato o la forma de utilizarlos, de manera que todas las herramientas que hacen uso de ellos sigan las mismas reglas, pudiendo compartir así los contenidos.

El estándar de LOM clasifica la información en diferentes categorías, proporciona una colección de metadatos que podemos asociar al LO, pero tienen un carácter descriptivo; es muy útil desde el punto de vista de su interpretación por las personas, pero es muy difícil montar algún sistema inteligente que utilice los LO's en función de sus metadatos o automatice algún control sobre ellos. Ideas de aplicaciones que automáticamente adaptan contenidos dependiendo de ciertos parámetros no son soportados con estos esquemas, deben ser cumplimentados con alguna estructura más que permita esta funcionalidad.

Esta estructura la forman las taxonomías y ontologías, que son clasificaciones de elementos que inciden más en el aspecto semántico de los mismos. Las ontologías son un elemento fundamental de la web semántica, suponen una capa por encima de RDF y los esquemas RDF. Son una descripción formal de los conceptos y las relaciones entre conceptos. El uso de ontologías en el desarrollo de Sistemas de Información permite establecer correspondencia y relaciones entre los diferentes dominios de entidades de información. Las ontologías toman un papel clave en la resolución de interoperabilidad semántica entre sistemas de información y su uso. La relación de la web semántica y el aprendizaje asistido por computador.

1.7 - La web semántica en e-learning

La web semántica es la tecnología que pretende proporcionar significado a los contenidos web. Para hacerlo se propone utilizar algún tipo de extensión de la propia tecnología web que permita clasificar por su significado y de forma automática los contenidos de los diferentes datos. Y todo esto se propone hacerlo sin utilizar técnicas de centralización de los datos y sin intervención de ninguna autoridad competente que autorice o verifique la clasificación.

Como herramientas principales de la web semántica se constituyen principalmente: un sistema de descripción de recursos (localizados por su URI) mediante un lenguaje estandarizado (RDF) y ontologías de contenidos que permiten especificar explícitamente los conceptos de un dominio concreto, sus propiedades y sus relaciones (entre ellos y entre conceptos de otros dominios).

Además de estas herramientas, son necesarios otros componentes, los llamados mediadores educativos, que se encargan de realizar las tareas relacionadas con la recepción de las diferentes consultas y su propagación a los distintos nodos educativos a los que pertenecía la información consultada. Su principal tarea, por tanto es la integración de la información, actuando como intermediarios entre las aplicaciones y los repositorios así como representando datos heterogéneos de forma homogénea y estructura. Son los mediadores los encargados de proporcionar al usuario del sistema un entorno virtual y globalizado, a partir de unos recursos dispersos y heterogéneos pero clasificados e indexados por su significado.

1.8 – Descripción RDF. Manual de sintaxis RDF

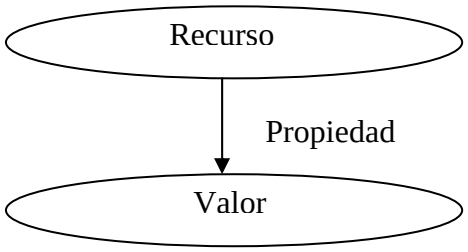
1.8.1 - Un vistazo a RDF

RDF (Resource Description Framework) es un *framework*, es decir, una estructura para intercambiar metadatos. RDF maneja fundamentalmente tres elementos:

- Recurso: Es lo que vamos a describir, en nuestro caso nuestra página web.
- Propiedad: el autor de una página, la fecha de creación... pero no el valor de los mismos.
- Sentencia: Es la combinación de los dos anteriores dándole un valor a la propiedad. “El **autor** de **http://www.textuality.com/RDF.why.html** es **Tim Bray**”

a. Visión general de la sintaxis de RDF

Como hemos dicho antes RDF se compone de sentencias con la siguiente forma:



Donde en el ejemplo anterior el recurso sería la página web, la propiedad sería autor y el valor sería Tim Bray.

Vamos a ver un ejemplo para comprender la estructura general del RDF:

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.             xmlns:exterms="http://www.example.org/terms/">

4.   <rdf:Description rdf:about="http://www.example.org/index.html">
5.       <exterms:creation-date>August 16, 1999</exterms:creation-
date>
6.   </rdf:Description>

7. </rdf:RDF>
```

Podemos separar este código en dos partes un cuerpo y un envoltorio para describir qué cosas vamos a utilizar dentro. El envoltorio sería

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.             xmlns:exterms="http://www.example.org/terms/">


7. </rdf:RDF>
```

Mientras que el cuerpo sería:

```
<rdf:Description rdf:about="http://www.example.org/index.html">
5.       <exterms:creation-date>August 16, 1999</exterms:creation-date>
6.   </rdf:Description>
```

Del envoltorio sacamos que <?xml version="1.0"?> es la declaración de XML que indica que lo que sigue es xml y que versión es.

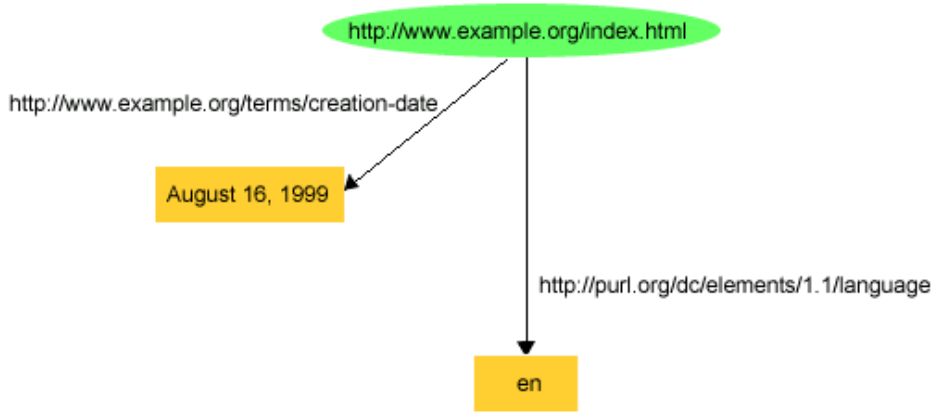
A continuación vemos entre las etiquetas <rdf:RDF ...> </rdf:RDF> todo el código que representa lo que es el RDF en sí.
Dentro de la etiqueta <rdf:RDF ...> definimos en que lugar estarán definidas las etiquetas que utilizaremos en el cuerpo. En este caso se utilizarían dos tipos de etiquetas:

- Las etiquetas contenidas en http://www.w3.org/1999/02/22-rdf-syntax-ns#.
- Las etiquetas contenidas en http://www.example.org/terms/

Si nos metemos en la primera url se nos descargará un fichero con todas las etiquetas que podremos utilizar, siempre precedidas por la etiqueta que en dicho archivo se nos indique en el primer caso es rdf: y en el segundo caso es exterms: Como vemos unas líneas más abajo en nuestro ejemplo utilizamos la etiqueta Description de la primera url y aparece precedida por rdf: y a continuación utilizamos la etiqueta creation-date de la segunda url precedido por exterms:

Pero veamos antes como se utilizan las URIs para hacer esas descripciones y cómo se pueden abreviar para no tener que escribirlas enteras cada vez que las utilicemos, ya que por ejemplo para hacer la siguiente descripción resultaría demasiado largo:

http://www.example.org/index.html has a **creation-date** whose value is **August 16, 1999**
http://www.example.org/index.html has a **language** whose value is **English**



Suponiendo que creation-date fuese una propiedad definida en http://www.example.org/terms y que language es una propiedad definida en http://purl.org/dc/elements/1.1/ Quedaría:

<http://www.example.org/index.html> <http://www.example.org/terms/creation-date> "August 16, 1999"
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/language> “en”

Esto significaría que cada vez que fuésemos a utilizar una de estas propiedades tendríamos que usar todo el link para evitarlo podemos asignarle una abreviatura más corta con la que referirnos a ellas. Algunas de estas abreviaciones son:

prefix `rdf:`, namespace URI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#`
prefix `rdfs:`, namespace URI: `http://www.w3.org/2000/01/rdf-schema#`
prefix `dc:`, namespace URI: `http://purl.org/dc/elements/1.1/`
prefix `owl:`, namespace URI: `http://www.w3.org/2002/07/owl#`
prefix `ex:`, namespace URI: `http://www.example.org/` (or `http://www.example.com/`)
prefix `xsd:`, namespace URI: `http://www.w3.org/2001/XMLSchema#`

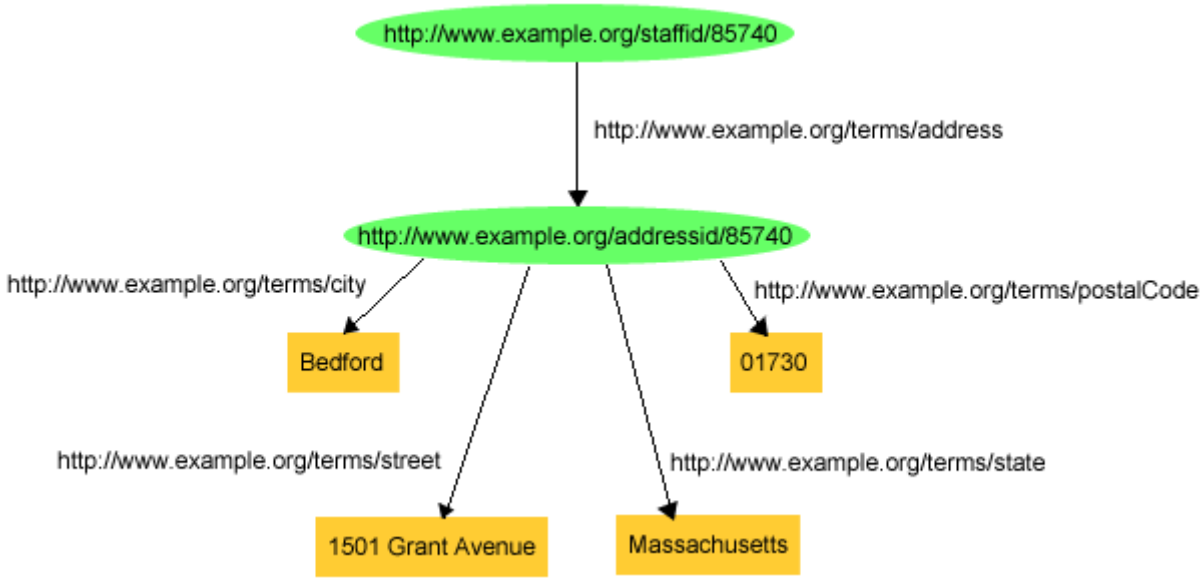
Con lo que nuestro ejemplo quedaría mucho más reducido:

ex:index.html exterms:creation-date "August 16, 1999"
ex:index.html dc:language "en"

Esto se correspondería entonces con la definición en el envoltorio de nuestro código rdf ahí meteremos las referencias a todos los vocabularios que vamos a utilizar y sus abreviaturas que tendremos que poner siembre delante de la etiqueta que vayamos a utilizar.

Vamos a introducir ahora el concepto de nodo en blanco. Hay determinadas estructuras que nos pueden surgir en rdf que hagan que tengamos demasiadas URIs intermedias que realmente no son necesarias y a las que les asignaremos esos nodos en blanco.
Por ejemplo si queremos hablar de la dirección del autor de una página web la podemos poner como un string seguido de calle número y piso o puede interesarnos más desglosarlo en varios nodos uno correspondiéndose con la calle, otro con el número, otro con el piso...

Eso puede dar como resultado estructuras como esta



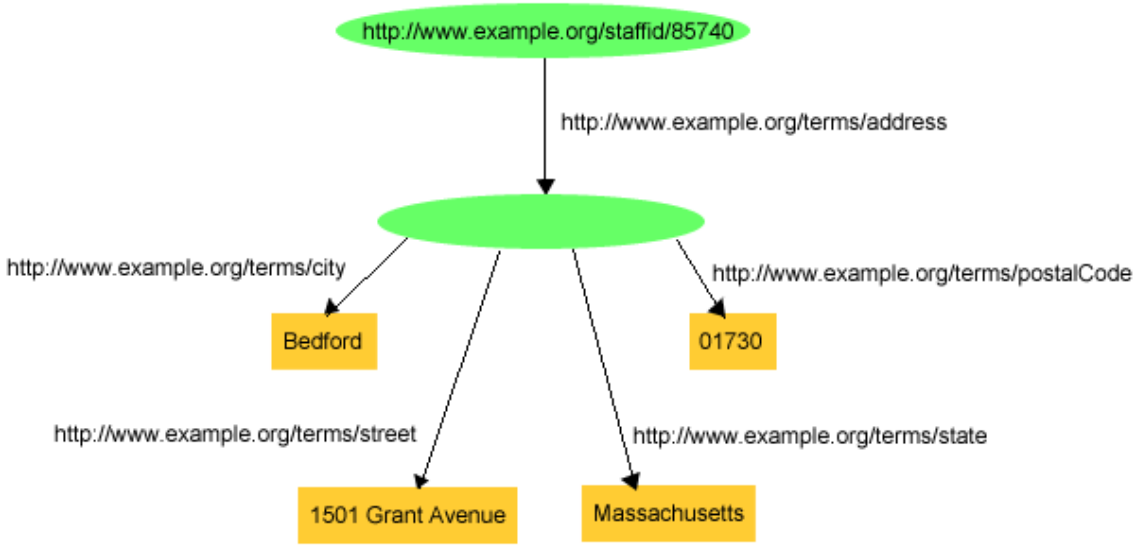
Donde la dirección http://www.example.org/staffid/85740 representa un empleado de una empresa que se corresponde con el número 85740 y cuya dirección queremos detallar desglosada. ¿Realmente necesitamos el nodo

http://www.example.org/addressid/85740? ¿O podemos prescindir de él ya que no nos aporta ninguna información en concreto?

Si no prescindimos de ese nodo la estructura nos quedaría como:

exstaff:85740	externs:address	exaddressid:85740
exaddressid:85740	externs:street	"1501 Grant Avenue"
exaddressid:85740	externs:city	"Bedford"
exaddressid:85740	externs:state	"Massachusetts"
exaddressid:85740	externs:postalCode	“3532”

Esto nos obliga a definir un nuevo concepto *address* que realmente no es necesario, eso es lo que llamaríamos una URI intermedia no necesaria. Por lo tanto podría ser un nodo en blanco y quedar como sigue.



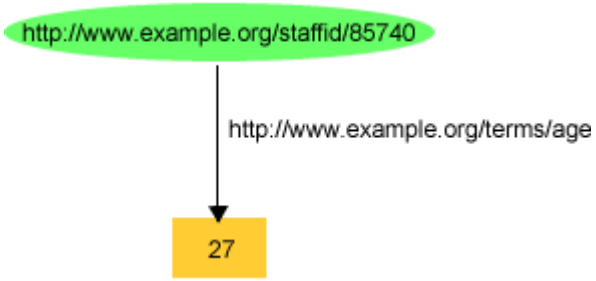
Pero hay que encontrar una solución para unir todos los elementos que conforman la dirección (calle, número... etc) porque sino la descripción queda suelta:

exstaff:85740	externs:address	????????????
????????????	externs:street	"1501 Grant Avenue"
????????????	externs:city	"Bedford"
????????????	externs:state	"Massachusetts"
????????????	externs:postalCode	“3532”

Para que esto no quede así y para poder diferenciar nodos en blanco entre sí tenemos que introducir un nuevo concepto, el de los identificadores de nodos en blanco que se forman con _:nombre quedando

exstaff:85740 exterms:address _:johnaddress
_:johnaddress exterms:street "1501 Grant Avenue"
_:johnaddress exterms:city "Bedford"
_:johnaddress exterms:state "Massachusetts"
_:johnaddress exterms:postalCode “3532”

Vamos a introducir ahora un nuevo concepto, los literales (*Typed Literals*).
El problema se plantea cuando queremos dar un dato como por ejemplo una edad y se tiene que interpretar de una forma determinada, es decir, John Smith tiene 27 años se podría representar como:



Pero ese 27 podría interpretarse como un string compuesto por un 2 y un 7, o como un número en base 8... pero queremos que se interprete como el número decimal 27. Esto se soluciona en otros lenguajes mediante tipos integer, string...y en RDF mediante los *typed literals*. Un *typed literal* está formado por la asociación de un string a una URI que identifica un determinado tipo. Esto representa un solo nodo en los gráficos.

Por ejemplo el caso anterior sería:

<http://www.example.org/staffid/85740>
<http://www.example.org/terms/age>
"27"^^<http://www.w3.org/2001/XMLSchema#integer>

que en versión abreviada sería:

exstaff:85740 exterms:age "27"^^xsd:integer

Similar a esto se podría hacer el ejemplo de la fecha con un tipo date. Antes de seguir hay que tener en cuenta que los “tipos” utilizados en RDF son los tipos de las URIs a las que hagamos referencia, es decir, RDF no tiene tipos construidos internamente como otros lenguajes pueden tener el Integer o el String.

La utilización de dichos *datatypes* se hace mediante la etiqueta rdf:datatype como se utiliza en el ejemplo siguiente para representar el tipo fecha:

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.             xmlns:exterms="http://www.example.org/terms/">

4.   <rdf:Description rdf:about="http://www.example.org/index.html">
5.     <exterms:creation-date rdf:datatype=
         "http://www.w3.org/2001/XMLSchema#date">1999-08-16
       </exterms:creation-date>
6.   </rdf:Description>

7. </rdf:RDF>
```

Aquí vemos otro caso en el que la URI que referencia el tipo no puede ser abreviado. En general las URIs que se utilizan en las etiquetas no pueden ser abreviadas. Pero aunque RDF no nos facilita esa posibilidad podemos utilizar las entidades de xml para abreviar dichas URIs como se hace en el siguiente ejemplo donde la URI que representa el tipo date se abrevia por xsd

```
1. <?xml version="1.0"?>
2. <!DOCTYPE rdf:RDF [<!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#">]>

3. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.             xmlns:exterms="http://www.example.org/terms/">

5.   <rdf:Description rdf:about="http://www.example.org/index.html">
6.     <exterms:creation-date rdf:datatype="&xsd;date">1999-08-16
       </exterms:creation-date>
7.   </rdf:Description>

8. </rdf:RDF>
```

Todas las descripciones de un mismo elemento las meteremos dentro del mismo bloque como se muestra a continuación:

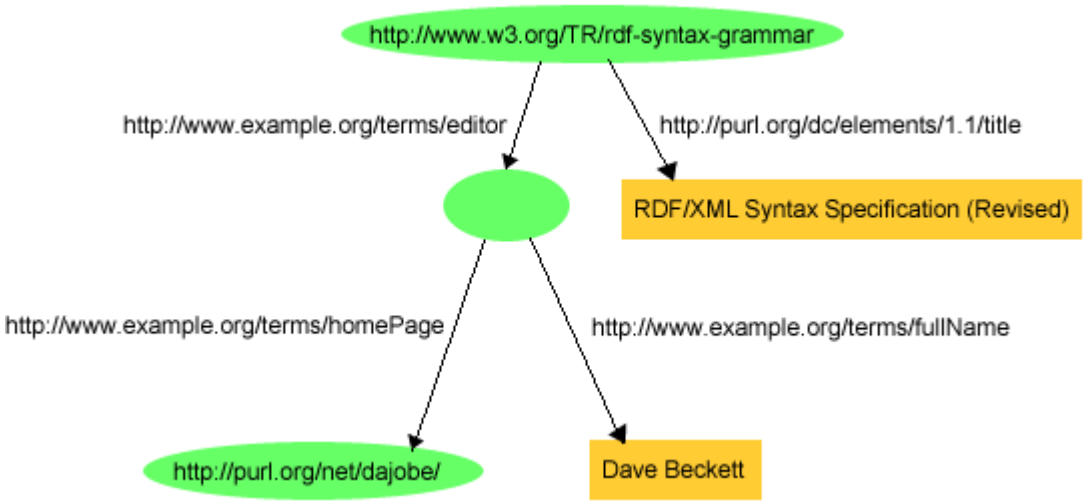
```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.              xmlns:dc="http://purl.org/dc/elements/1.1/"
4.              xmlns:exterms="http://www.example.org/terms/">

5.    <rdf:Description rdf:about="http://www.example.org/index.html">
6.         <exterms:creation-date>August 16, 1999</exterms:creation-
date>
7.         <dc:language>en</dc:language>
8.         <dc:creator
rdf:resource="http://www.example.org/staffid/85740"/>
9.    </rdf:Description>
```

En la línea 8 de este ejemplo vemos un concepto nuevo. Cuando el valor de una declaración es el contenido de una URI tenemos que ponerlo como se muestra arriba con la etiqueta rdf:resource ya que si lo pusiéramos así:

<dc:creator>http://www.example.org/staffid/85740</dc:creator>
Se tomaría como valor del creador la cadena literal que representa la URI y no su contenido que es realmente lo que queremos. En este caso no se pueden utilizar abreviaturas para referirse a dicha URI.

Veamos ahora como se representaría en RDF un nodo en blanco como el de la siguiente figura:




```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.              xmlns:dc="http://purl.org/dc/elements/1.1/"
4.              xmlns:exterms="http://example.org/stuff/1.0/">

5.     <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-
grammar">
6.       <dc:title>RDF/XML Syntax Specification (Revised)</dc:title>
7.       <exterms:editor rdf:nodeID="abc"/>
8.     </rdf:Description>

9.     <rdf:Description rdf:nodeID="abc">
10.       <exterms:fullName>Dave Beckett</exterms:fullName>
11.       <exterms:homePage
rdf:resource="http://purl.org/net/dajobe/"/>
12.    </rdf:Description>

13. </rdf:RDF>
```

La solución como se ve en el ejemplo es ponerle un nombre al nodo mediante la etiqueta rdf:nodeID en este caso rdf:nodeID=”abc”

Ya hemos visto en general lo más importante de la sintaxis de RDF pero vamos a ver otras capacidades aparte.

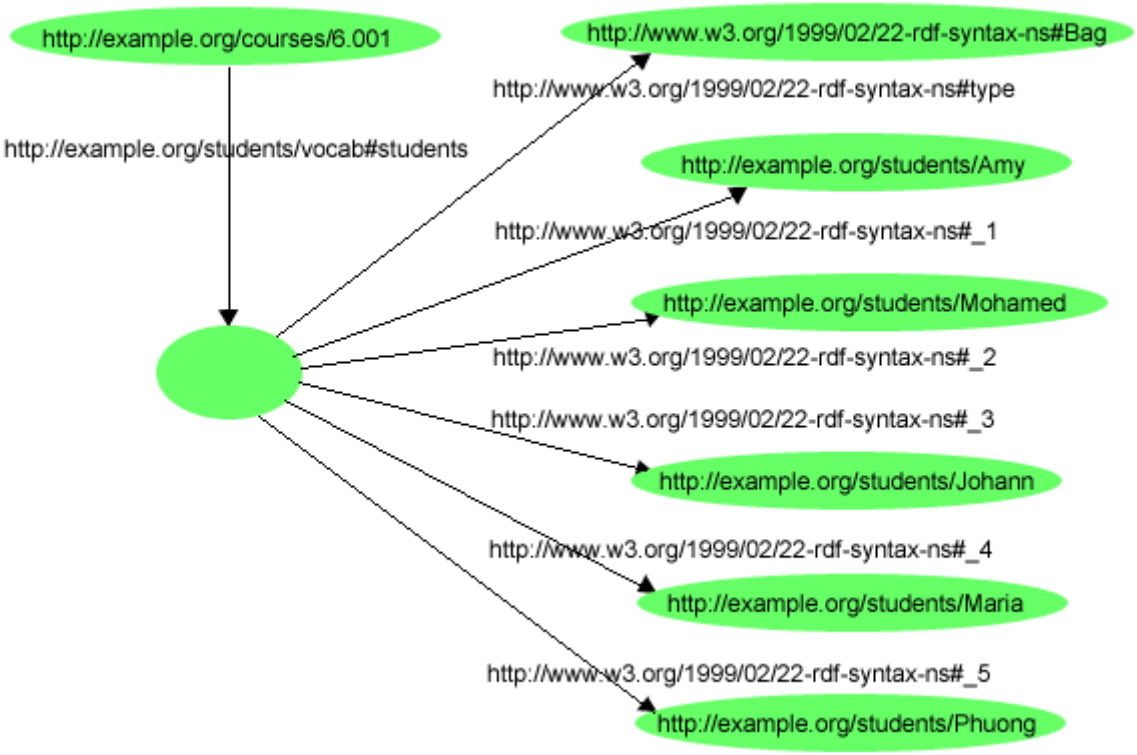
b.Otras capacidades de RDF
b.1 Los contenedores

Si necesitamos representar un grupo de cosas como por ejemplo una página web que tiene varios autores podemos utilizar tres tipos de contenedores que son:

- rdf:Bag -> Puede haber miembros duplicados y el orden no importa.
- rdf:Seq -> Puede haber miembros duplicados y el orden si importa.
- rdf:Alt -> Representa alternativas, como por ejemplo, si damos el título de un libro y queremos dar el mismo titulo pero en ingles como alternativa.

Los miembros de estos conjuntos pueden ser definidos con una propiedad rdf:_n donde n es un entero distinto de cero (rdf:_1, rdf:_2, rdf:_2...)

Veamos un ejemplo de conjunto representamos la información contenida en “El curso 6001 tiene los estudiantes Amy, Mohamed, Johan, María y Phuong”



```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

   <rdf:Description rdf:about="http://example.org/courses/6.001">
      <s:students>
         <rdf:Bag>
            <rdf:li rdf:resource="http://example.org/students/Amy"/>
            <rdf:li rdf:resource="http://example.org/students/Mohamed"/>
            <rdf:li rdf:resource="http://example.org/students/Johann"/>
            <rdf:li rdf:resource="http://example.org/students/Maria"/>
            <rdf:li rdf:resource="http://example.org/students/Phuong"/>
         </rdf:Bag>
      </s:students>
   </rdf:Description>
</rdf:RDF>
```

En este ejemplo se utiliza la etiqueta rdf:li para que sea nemotécnica con list item para evitar tener que numerar los elementos explícitamente ya que aquí el orden nos da igual.

NOTA: La utilización de los conjuntos descritos arriba son solo una sugerencia, se pueden hacer grupos " a mano" haciendo que todos los miembros de ese grupo cuelguen de un nodo creado por nosotros.

En este ejemplo anterior vemos una forma diferente de expresar las propiedades más abreviadas que se explica a continuación. Veamos las diferencias entre los dos siguientes ejemplos.

Ejemplo1

```
1.   <?xml version="1.0"?>
2.   <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
3.   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.               xmlns:exterms="http://www.example.com/terms/"
5.               xml:base="http://www.example.com/2002/04/products">

6.     <rdf:Description rdf:ID="item10245">
7.          <rdf:type rdf:resource="http://www.example.com/terms/Tent"/>
8.          <exterms:model
rdf:datatype="&xsd;string">Overnighter</exterms:model>
9.          <exterms:sleeps rdf:datatype="&xsd;integer">2</exterms:sleeps>
10.         <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
11.         <exterms:packedSize
rdf:datatype="&xsd;integer">784</exterms:packedSize>
12.    </rdf:Description>

  ...other product descriptions...

13.  </rdf:RDF>
```

Ejemplo2

```
1.   <?xml version="1.0"?>
2.   <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
3.   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.               xmlns:exterms="http://www.example.com/terms/"
5.               xml:base="http://www.example.com/2002/04/products">

6.     <exterms:Tent rdf:ID="item10245">
7.          <exterms:model
rdf:datatype="&xsd;string">Overnighter</exterms:model>
8.          <exterms:sleeps rdf:datatype="&xsd;integer">2</exterms:sleeps>
9.          <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
10.         <exterms:packedSize
rdf:datatype="&xsd;integer">784</exterms:packedSize>
11.    </exterms:Tent>

  ...other product descriptions...

12.  </rdf:RDF>
```

En los dos ejemplos anteriores se están enumerando las características de una tienda “Tent”. En el ejemplo 1 utilizamos la etiqueta Description y a continuación definiendo el rdf:type como venimos haciendo desde el principio pero en el segundo se utiliza una abreviación.
En RDF es común que la fuentes que se utilizan (los links) tengan propiedades rdf:type que describen dichas fuentes como determinados tipos o clases. Estas fuentes se llaman *typed node elements* para los que se puede utilizar una abreviación especial como en los ejemplos anteriores. Donde la propiedad rdf:type y su valor se quitan y sustituimos el rdf:Description por el nombre correspondiente al valor del rdf:type que hemos quitado, quedando como en el ejemplo 2.

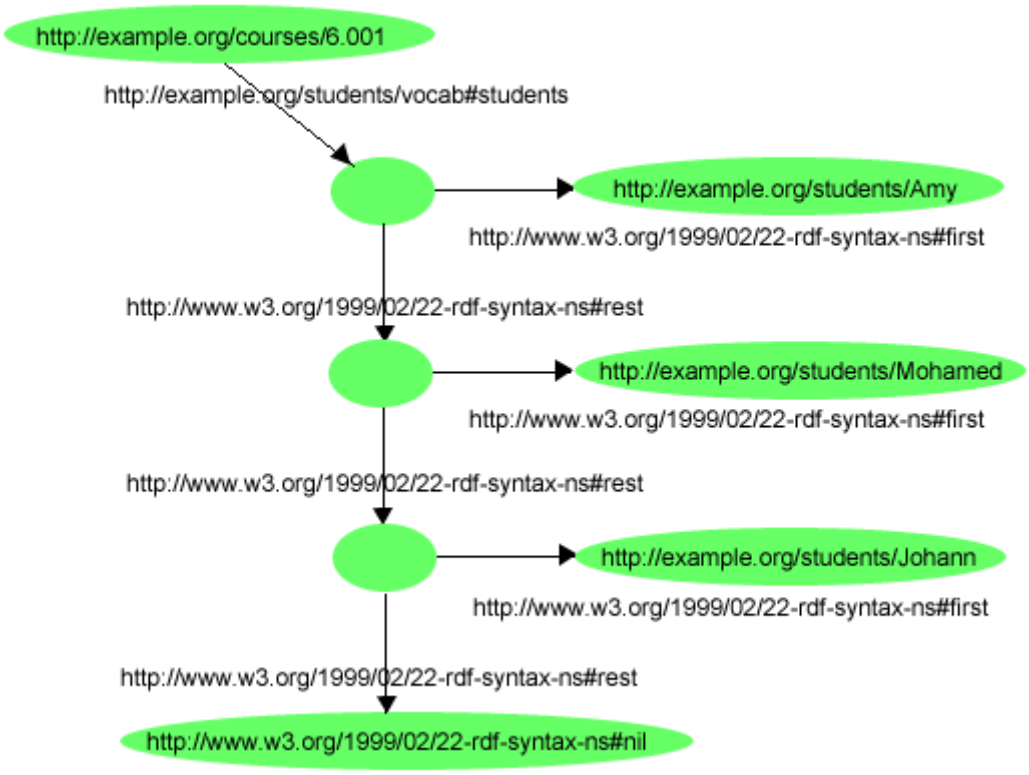
Una fuente puede estar descrita por más de un rdf:type, solo uno de ellos se puede abreviar de la forma anterior, el resto tienen que aparecer como en el ejemplo 1.

Luego queda en manos del programador el diferenciar qué estructuras le vienen mejor según el significado de la información que está manejando.

b.2 Colecciones

El problema de las construcciones anteriores es que dejan la posibilidad abierta de que esos grupos tengan más miembros, es decir, en ningún sitio se indica que son esos y solo esos los miembros del conjunto. Para solucionar ese tema están las colecciones o “Collections” que se definen como una lista con la etiqueta rdf:List con las propiedades predefinidas de rdf:first, rdf:rest y con la fuente predefinida rdf:list.

Veamos un ejemplo, representemos la frase “Los estudiantes del curso 6.001 son Amy, Mohamed y Johann”



En este gráfico, cada miembro de la colección es el valor de una propiedad rdf:first cuyo recurso es una fuente (o un nodo en blanco como en este caso) que representa una lista. Esta lista está unida al resto de listas con la propiedad rdf:rest. El final de la lista se señala con la propiedad rdf:rest señalando a rdf:nil. El lenguaje RDF define las propiedades rdf:first y rdf:rest como del tipo rdf:List así que la información del hecho de que estos nodos son listas puede ser inferida y no es necesario declararlo con un rdf:type en cada nodo.

Para definir una colección tenemos que utilizar otra etiqueta nueva, rdf:parseType que para definir una colección se utilizaría como sigue:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

   <rdf:Description rdf:about="http://example.org/courses/6.001">
      <s:students rdf:parseType="Collection">
            <rdf:Description
rdf:about="http://example.org/students/Amy"/>
            <rdf:Description
rdf:about="http://example.org/students/Mohamed"/>
            <rdf:Description
rdf:about="http://example.org/students/Johann"/>
      </s:students>
   </rdf:Description>
</rdf:RDF>
```

Pero hay que tener en cuenta que todos estos formatos son sencillamente sugerencias de cómo podemos organizar nuestros datos. El programador podrá crear sus estructuras utilizando el rdf:first y el rdf:rest como crea conveniente sin tener que utilizar explícitamente una colección. El ejemplo anterior sin colecciones y formando una lista quedaría como sigue:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:s="http://example.org/students/vocab#">

<rdf:Description rdf:about="http://example.org/courses/6.001">
   <s:students rdf:nodeID="sch1"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch1">
   <rdf:first rdf:resource="http://example.org/students/Amy"/>
   <rdf:rest rdf:nodeID="sch2"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch2">
   <rdf:first rdf:resource="http://example.org/students/Mohamed"/>
   <rdf:rest rdf:nodeID="sch3"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch3">
   <rdf:first rdf:resource="http://example.org/students/Johann"/>
   <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-
ns#nil"/>
</rdf:Description>
</rdf:RDF>
```

La posibilidad de hacer nuestras listas a mano tiene ventajas e inconvenientes, la ventaja es que si queremos hacer una estructura que sea parecida a una colección pero no exactamente igual, podremos hacerlo a mano fácilmente. Lo malo es que sin darnos cuenta podemos construir algo que no tenga sentido o no esté bien hecho, mientras que si hacemos una colección sabemos que va a estar bien.
Si la aplicación que va a leer el RDF espera una colección bien formada habría que añadirle a la misma una funcionalidad que haga de compilador, es decir, que se mire el código que implementa la colección para asegurarse de que eso está bien escrito y tenga sentido, eso hará que nuestro sistema sea más robusto.

b.3 Reificación

Este nuevo concepto nos ofrece la posibilidad de añadir información a una sentencia de las que hemos utilizado hasta ahora. Es decir, si creamos una sentencia “El autor del libro es Pepito” y queremos asociarle información cuando se declaró dicha sentencia, quién la declaró... etc

Vamos a intentar explicarlo con un ejemplo. Recordemos el ejemplo que utilizamos más arriba en el que se hacía una sentencia sobre una tienda de campaña *Tent* que era un *item* de una tienda de deportes que tiene una URIref en la que hace un listado de todos los *items* que vende. Por lo tanto se referenciaba a dicha tienda mediante exproducts:item10245. Vamos a fijarnos concretamente en la sentencia que nos daba su peso:

exproducts:item10245 exterms:weight "2.4"^^xsd:decimal
Ahora imaginemos que queremos saber quién nos dio dicha información.

RDF nos ofrece un vocabulario para describir sentencias que es a lo que se llama reificación. Las palabras reservadas para esto son rdf:Statement, rdf:subject , rdf:predicate y rdf:object.

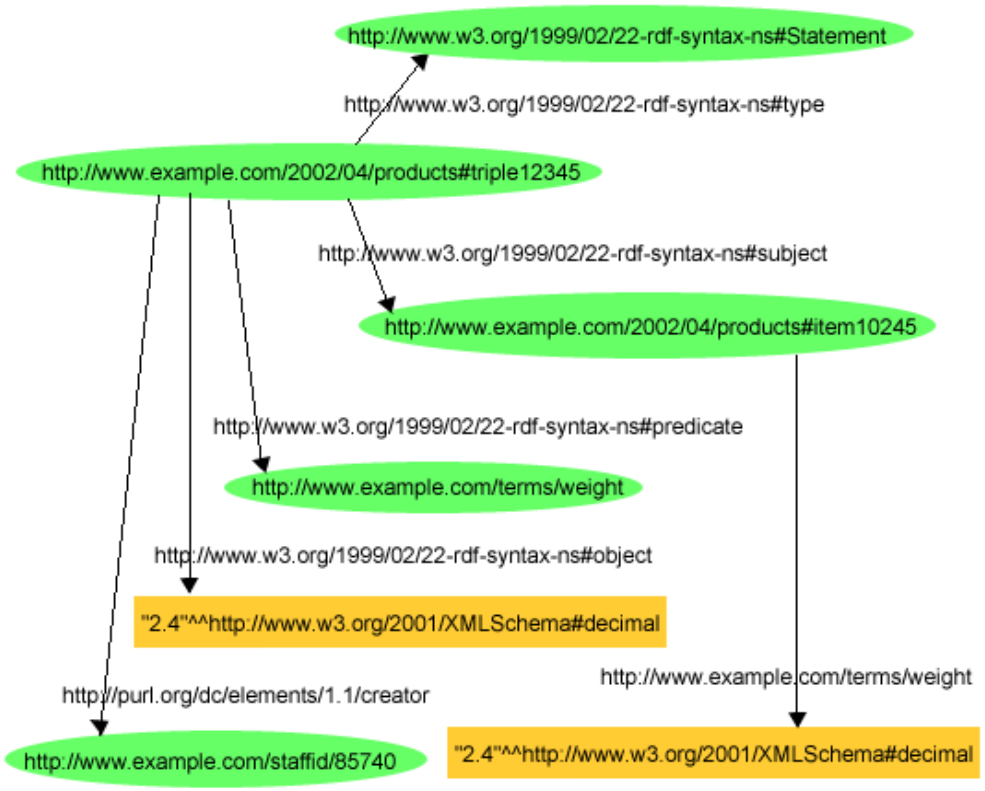
Si reificamos el ejemplo anterior de la tienda de campaña nos quedaría:

exproducts:triple12345	rdf:type	rdf:Statement
exproducts:triple12345	rdf:subject	exproducts:item10245
exproducts:triple12345	rdf:predicate	exterms:weight
exproducts:triple12345	rdf:object	"2.4"^^xsd:decimal

Esto quiere decir que la fuente identificada por la URIref exproducts:triple12345 es una sentencia RDF, que el *recurso* de la sentencia es la fuente identificada con exproducts:item10245, que la *propiedad* de la sentencia es la fuente identificada por exterms:weight y que el *valor* de la sentencia es el valor decimal definido por el *typed literal* "2.4"^^xsd:decimal.
Por lo tanto, si queremos decir que la información sobre la tienda de campaña nos la facilitó John Smith lo diríamos como se muestra a continuación:

exproducts:triple12345	rdf:type	rdf:Statement
exproducts:triple12345	rdf:subject	exproducts:item10245
exproducts:triple12345	rdf:predicate	exterms:weight
exproducts:triple12345	rdf:object	"2.4"^^xsd:decimal
exproducts:triple12345	dc:creator	exstaff:85740

Que se corresponde con el siguiente gráfico:



El código correspondiente al gráfico es:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:exterms="http://www.example.com/terms/"
            xml:base="http://www.example.com/2002/04/products">

  <rdf:Description rdf:ID="item10245">
     <exterms:weight rdf:datatype="&xsd;decimal">2.4</exterms:weight>
  </rdf:Description>

  <rdf:Statement rdf:about="#triple12345">
     <rdf:subject
rdf:resource="http://www.example.com/2002/04/products#item10245"/>
     <rdf:predicate
rdf:resource="http://www.example.com/terms/weight"/>
     <rdf:object rdf:datatype="&xsd;decimal">2.4</rdf:object>
     <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
  </rdf:Statement>

</rdf:RDF>
```

A continuación vemos una forma de llevar a cabo una reificación con la etiqueta rdf:ID que ya estudiamos en su momento quedando de la siguiente manera.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xmlns:exterms="http://www.example.com/terms/"
            xml:base="http://www.example.com/2002/04/products">

  <rdf:Description rdf:ID="item10245">
     <exterms:weight rdf:ID="triple12345"
rdf:datatype="&xsd;decimal">2.4
     </exterms:weight>
  </rdf:Description>

  <rdf:Description rdf:about="#triple12345">
     <dc:creator rdf:resource="http://www.example.com/staffid/85740"/>
  </rdf:Description>

</rdf:RDF>
```

En este caso especificando el atributo rdf:ID=”triple12345” en el elemento exterms:weight obtenemos:

exproducts:item10245 exterms:weight "2.4"^^xsd:decimal

Mas la reificación

exproducts:triple12345	rdf:type	rdf:Statement
exproducts:triple12345	rdf:subject	exproducts:item10245
exproducts:triple12345	rdf:predicate	exterms:weight
exproducts:triple12345	rdf:object	"2.4"^^xsd:decimal

Declarar la reificación no es lo mismo que declarar la sentencia original y ninguna de las dos implica a la otra. Es decir, cuando alguien dice que John Smith declara algo sobre el peso de la tienda de campaña no está haciendo una sentencia sobre el peso de la tienda de campaña, sino que está haciendo una sentencia sobre algo que ha dicho John Smith. Así mismo cuando alguien describe el peso de la tienda de campaña no está haciendo también una sentencia sobre lo que dijo John Smith.

b.4 Valores estructurados: rdf:Value

Hasta ahora solo hemos utilizado relaciones binarias del tipo “Pepe es el autor del libro La Sombra”. El único caso en el que nos hemos enfrentado a una relación n-aria fue en el caso de desglosar la dirección de John en calle, número, ciudad... Caso que solucionamos con el nodo en blanco.

El problema surge cuando a uno de los datos le queremos dar más importancia que a los demás, por ejemplo, si en la dirección le quisiéramos dar más importancia a la calle que al número o la ciudad.

O por ejemplo, en el caso de la tienda de campaña, al dar el peso estaría mejor si en vez de decir que el peso es un número decimal representado por 24 dijésemos que el peso de la tienda son 24 kilogramos. Y entre la información de 24 y kilogramos le queremos dar más importancia al 24.

Con el atributo rdf:value le damos más importancia al dato que nosotros queramos. Quedando como se muestra a continuación:

exproduct:item10245	exterms:weight	_:weight10245
_:weight10245	rdf:value	"2.4"^^xsd:decimal
_:weight10245	exterms:units	exunits:kilograms

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description
rdf:about="http://www.example.com/2002/04/products#item10245">
     <exterms:weight rdf:parseType="Resource">
       <rdf:value rdf:datatype="&xsd;decimal">2.4</rdf:value>
       <exterms:units
rdf:resource="http://www.example.org/units/kilograms"/>
     </exterms:weight>
  </rdf:Description>

</rdf:RDF>
```

En este ejemplo se muestra una nueva forma de utilizar el atributo rdf:parseType utilizado con anterioridad en este documento. En este caso rdf:parseType="Resource". Dicho atributo se utiliza para indicar que el contenido de un elemento tiene que ser interpretado como una descripción de una nueva fuente sin tener que escribir un elemento rdf:Description anidado. En este caso el atributo rdf:parseType="Resource" utilizado en la propiedad exterms:weight indica que un nodo en blanco tiene que ser creado como valor de la propiedad exterms:weight, y que los elementos rdf:value y exterms:units describen propiedades de ese nodo en blanco.

b.5 XML Literals

Hay veces que el valor de una propiedad tiene que ser un fragmento de XML, o texto con algún formato o marcado determinado. Por ejemplo los títulos de un libro de matemáticas pueden contener formulas que se representen en MathML.

RDF/XML ofrece una notación especial para facilitar estos casos. Esto se soluciona también con el atributo rdf:parseType. Dándole a un elemento el atributo rdf:parseType="Literal" se indica que el contenido del elemento tiene que ser interpretado como un fragmento de XML. Veamos un ejemplo:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xml:base="http://www.example.com/books">

  <rdf:Description rdf:ID="book12345">
     <dc:title rdf:parseType="Literal">
       <span xml:lang="en">
         The <em>&lt;br /&gt;</em> Element Considered Harmful.
       </span>
     </dc:title>
  </rdf:Description>

</rdf:RDF>
```

c. Definiendo Vocabularios RDF: Esquemas RDF

Ya hemos visto como se utiliza el RDF. Pero a lo largo de todos los ejemplos hemos ido utilizando un vocabulario incluido en ciertas URIs que acortábamos dándole una especie de nick más fácil de utilizar. Pero ¿cómo se forman esos vocabularios? ¿Cómo podemos crear nosotros esas propiedades para luego usarlas en nuestras descripciones? Eso es lo que se hace con los esquemas RDF (o RDF schema). Por tanto, dicho esquema nos dará la posibilidad de definir o crear una serie de propiedades y de determinar como se relacionarán entre sí, es decir, el esquema de RDF nos provee de un sistema de tipos (*Type System*) para RDF. En cierto modo se parece a la programación de tipos en programación orientada a objetos. Ya que nos permite declarar una fuente como una instancia de una o más clases. Es más, permite organizar dichas clases de forma jerárquica.

Las prestaciones del esquema RDF se presentan como un vocabulario RDF a su vez, es decir, como un concreto conjunto de fuentes con sus significados concretos. Las fuentes utilizadas en el esquema RDF son las URIs con el prefijo http://www.w3.org/2000/01/rdf-schema# (asociado con el Qname rdfs:). Los vocabularios descritos con el esquema RDF son gráficos RDF correctamente formados. Por lo tanto, las herramientas construidas de forma que no interpreten

también el vocabulario del esquema RDF podrán interpretar un esquema como un gráfico RDF bien formado consistente en varias fuentes y propiedades, pero no "entenderá" la información adicional de los términos del esquema RDF. Para entender esta información extra, las herramientas de RDF tienen que ser capaces de procesar y extender lenguajes que incluyan no solo el vocabulario rdf: sino también el vocabulario rdfs: junto con sus significados. Discutiremos esto más a fondo más adelante.

c.1 Describiendo clases

Las clases en RDF schema será lo que vamos a describir, más o menos funcionan igual que las clases en Java. Se describen utilizando las fuentes que nos proporciona el RDF schema rdfs:Class, rdfs:Resource y las propiedades rdf:type y rdfs:subClassof.

Supongamos que hay una compañía example.org que quiere utilizar RDF para dar información sobre diferentes tipos de vehículos a motor. En el esquema RDF example.org necesitará primero una clase para representar la categoría de cosas que son vehículos a motor. Las fuentes que pertenezcan a dicha clase serán sus instancias.

En el esquema RDF una clase es cualquier fuente que tenga una propiedad rdf:type cuyo valor sea la fuente rdfs:Class. La clase vehículo a motor se describiría asignándole una URIref por ejemplo ex:MotorVehicle (suponiendo que ex: es http://www.example.org/schema/vehicles) y describiendo dicha fuente con una propiedad rdf:type cuyo valor es la fuente rdf:Class.

Ex:MotorVehicle rdf:type rdfs:Class

Como ya hemos dicho con anterioridad, la propiedad rdf:type indica que una fuente es una instancia de una clase determinada. Teniendo ya descrito ex:MotorVehicle como una clase, la fuente exthings:companyCar debería ser descrito como un vehículo a motor mediante la sentencia RDF que sigue:

exthings:companyCar rdf:type ex:MotorVehicle

(Vemos que se sigue la regla de poner en mayúsculas las clases y en minúsculas las instancias de las clases)

Una fuente puede ser instancia de más de una clase.

Imaginemos que ahora se quieren definir clases específicas de vehículos, camiones, coches, camionetas...

ex:Van rdf:type rdfs:Class
ex:Truck rdf:type rdfs:Class

Pero para decir que estas clases son subclases de vehículo a motor sería como sigue

ex:Van rdfs:subClassOf ex:MotorVehicle

El significado de rdfs:subClassOf es que cualquier instancia de ex:Van es también una instancia de la clase ex:MotorVehicle. Por lo tanto si la fuente exthings:companyVan es una instancia de ex:Van entonces, la herramienta RDF podrá inferir que exthings:companyVan es también una instancia de

ex:MotorVehicle. Esto es a lo que nos referíamos antes con que la herramienta RDF que creemos deberá ser capaz de comprender los esquemas RDF para poder inferir este tipo de cosas, porque sino, si que será capaz de leer la última terna pero no será capaz de inferir la herencia.

La propiedad rdfs:subClassOf es transitiva.

Todas las clases definidas en RDF son subclases de la clase rdfs:Resource.

A continuación mostramos todos la jerarquía resultante de los ejemplos anteriores:

El esquema está simplificado para hacerlo más visible haciendo que no aparezca la propiedad rdf:type apuntando a una rdfs:Class.

El esquema anterior escrito en RDF sería:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">

<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="PassengerVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
```

```
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="MiniVan">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>
</rdf:RDF>
```

Como ya se explicó más arriba hay una forma de abreviar las descripciones de fuentes que tienen una propiedad rdf:type (*typed nodes*). Ya que los esquemas RDF son fuentes RDF, esta abreviación puede ser aplicada para la descripción de clases. Con dicha abreviación el ejemplo anterior quedaría:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">

<rdfs:Class rdf:ID="MotorVehicle"/>

<rdfs:Class rdf:ID="PassengerVehicle">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Truck">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Van">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="MiniVan">
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdfs:Class>

</rdf:RDF>
```

Esta abreviación será utilizada en el resto de esta sección.

En el ejemplo vemos que la forma de asignar un nombre a la nueva clase es con rdf:ID. Dicho nombre añadido a la URI asociada al xml:base dada en la cabecera nos dará la URI completa que se refiere a esa clase. Por ejemplo, la clase MiniVan será http://example.org/schemas/vehicles#MiniVan.

La forma de utilizar estas clases sería por ejemplo el declarar companyCar como una instancia de ex:MotorVehicle como sigue

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:ex="http://example.org/schemas/vehicles#"
  xml:base="http://example.org/things">
   <ex:MotorVehicle rdf:ID="companyCar"/>
</rdf:RDF>
```

c.2 Describiendo propiedades

Aparte de definir unas clases necesitaremos también las propiedades que describan esas clases. Dichas propiedades se declararán utilizando la clase rdf:Property, y las propiedades del esquema RDF rdfs:domain, rdfs:range y rdfs:subPropertyOF.

Todas las propiedades en RDF son instancias de la clase rdf:Property. Por lo tanto una nueva propiedad exterms:weightInKg se definirá como sigue:

exterms:weightInKg rdf:type rdf:Property

Las propiedades rdfs:range y rdfs:domain se utilizan para describir como las propiedades y las clases serán utilizadas junta para crear datos RDF.

La propiedad rdfs:range se utiliza para indicar que los valores de una propiedad concreta son instancias de una clase designada. Por ejemplo, si example.org quisiera indicar que la propiedad ex:autor tiene valores que son instancias de la clase ex:Person escribiría las sentencias RDF siguientes:

ex:Person rdf:type rdfs:Class
ex:author rdf:type rdf:Property
ex:author rdfs:range ex:Person

Esta sentencia declara que ex:Person es una clase, ex:Autor es una propiedad y que las sentencias que utilicen la propiedad ex:autor tiene instancias de ex:Person como valor.

Una propiedad puede tener cero, uno, o más rangos. Por ejemplo si la propiedad ex:hasMother no tiene ningún rango no se está diciendo nada, si tiene uno especificado como ex:Person estamos diciendo que el valor de esa propiedad es una persona, pero si tiene otro rango más por ejemplo ex:Female, la instancia que de valor a esa propiedad tiene que ser una persona y mujer. Es decir, tiene que ser una instancia de ambas clases.
Esto generaría dos sentencias:

ex:hasMother rdfs:range ex:Female
ex:hasMother rdfs:range ex:Person

La propiedad rdfs:range también se puede utilizar para indicar que el valor de una propiedad viene determinado por un *typed literal*. Por ejemplo si example.org quiere indicar que la propiedad ex:age tiene valores del tipo xsd:integer, escribiría las siguientes sentencias:

ex:age	rdf:type	rdf:Property
ex:age	rdfs:range	xsd:integer

El tipo xsd:integer viene identificado por su URIref (siendo la URIref completa http://www-w3.org/2001/XMLSchema#integer). Esta URIref puede ser usada sin declarar explícitamente en una sentencia que identifica un tipo de dato concreto. De todas formas, a menudo es conveniente declararlo explícitamente. Esto puede hacerse utilizando la clase rdfs:Datatype del RDF esquema. Para declarar que xsd:integer es un tipo, example.org escribiría la siguiente sentencia:

xsd:integer	rdf:type	rdfs:Datatype

Esto solo sirve para aclarar que existe ese tipo, no es ni una nueva declaración, ni se puede hacer una nueva declaración de tipos fuera de los tipos que nos ofrece RDF.

La propiedad rdfs:domain se utiliza para indicar que una propiedad en particular se aplica a una determinada clase. Por ejemplo si example.org quisiera indicar que la propiedad ex:autor se aplica a instancias de la clase ex:Book, escribiría la siguiente sentencia:

ex:Book	rdf:type	rdfs:Class
ex:author	rdf:type	rdf:Property
ex:author	rdfs:domain	ex:Book

Estas sentencias declaran que book es una clase que tiene como propiedad author.

Una propiedad determinada, pueda tener cero, una o más propiedad de dominio. Si no tiene ninguna es que cualquier clase puede tener esa propiedad. Y si tiene más pues que cada una de esas clases tiene esa propiedad.

El uso de estas dos propiedades range y domain puede entenderse mejor extendiendo el ejemplo del vehículo a motor, añadiéndole al esquema dos nuevas propiedades; ex:registeredTo y ex:rearSeatLegRoom, una nueva clase ex:Person y describiendo explícitamente el tipo xsd:integer. La propiedad ex:registeredTo se aplica a cualquier instancia de la clase ex:MotorVehicle y su valor es de la clase ex:Person. Sin embargo, ex:rearSeatLegRoom solo es aplicable a instancias de la clase ex:PassengerVehicle. El valor es un xsd:integer dando el número de centímetros de espacio para las piernas en el vehículo. Todo esto se muestra en el siguiente ejemplo:

```
<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="rearSeatLegRoom">
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>

<rdfs:Class rdf:ID="Person"/>

<rdfs:Datatype rdf:about="&xsd;integer"/>
```

Nota: la etiqueta <rdf:RDF> no se utiliza aquí porque se supone que este trozo de código se incrustaría en el esquema del vehículo dado más arriba.

El esquema RDF ofrece la posibilidad de especializar propiedades así como las clases. Esta especialización de la relación entre dos propiedades se describe utilizando la propiedad predefinida rdfs:subPropertyOf. Por ejemplo, si ex:primaryDriver y ex:driver son dos propiedades, example.org puede describir estas propiedades y el hecho de que ex:primaryDriver es una especialización de ex:driver mediante las siguientes sentencias:

ex:driver	rdf:type	rdf:Property
ex:primaryDriver	rdf:type	rdf:Property
ex:primaryDriver	rdfs:subPropertyOf	ex:driver

El significado de esta relación rdfs:subPropertyOf es que si una instancia exstaff:fred es un ex:primaryDriver de la instancia ex:companyVan, entonces el esquema RDF define exstaff:fred también como un ex:driver de ex:companyVan. El código RDF/XML de este ejemplo es el que sigue:

```
<rdf:Property rdf:ID="driver">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Property>

<rdf:Property rdf:ID="primaryDriver">
  <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>
```

Una propiedad puede ser subpropiedad de cero, una o más propiedades. Cada propiedad rdfs:range y rdfs:domain que se aplica a una propiedad también lo hacen para cada una de sus subpropiedades. Por lo tanto en el ejemplo anterior, el esquema RDF define es:primaryDriver teniendo también un rdfs:domain de ex:MotorVehicle, debido a su relación de subpropiedad con respecto a ex:driver

A continuación se muestra todo el ejemplo de los vehículos al completo:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">

<rdfs:Class rdf:ID="MotorVehicle"/>

<rdfs:Class rdf:ID="PassengerVehicle">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Truck">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Van">
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="MiniVan">
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Person"/>

<rdfs:Datatype rdf:about="&xsd;integer"/>

<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="rearSeatLegRoom">
  <rdfs:domain rdf:resource="#PassengerVehicle"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>

<rdf:Property rdf:ID="driver">
  <rdfs:domain rdf:resource="#MotorVehicle"/>
</rdf:Property>
<rdf:Property rdf:ID="primaryDriver">
  <rdfs:subPropertyOf rdf:resource="#driver"/>
</rdf:Property>

</rdf:RDF>
```

Ahora veamos un ejemplo de como declararíamos instancias de las nuevas clases que hemos creado con el esquema RDF:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:ex="http://example.org/schemas/vehicles#"
            xml:base="http://example.org/things">

  <ex:PassengerVehicle rdf:ID="johnSmithsCar">
       <ex:registeredTo
rdf:resource="http://www.example.org/staffid/85740"/>
       <ex:rearSeatLegRoom
rdf:datatype="&xsd;integer">127</ex:rearSeatLegRoom>
       <ex:primaryDriver
rdf:resource="http://www.example.org/staffid/85740"/>
  </ex:PassengerVehicle>
</rdf:RDF>
```

d. Interpretando declaraciones de esquemas RDF

Veamos las diferencias entre una clase en java y una clase en RDF. Una clase en java tiene sus propiedades “metidas dentro” y son suyas nada más. Si una clase Libro tiene una propiedad autor es su propiedad que nada tendrá que ver con la propiedad autor de otra clase que sea por ejemplo Software.
En RDF las propiedades se declaran aparte de las clases y pueden ser comunes a varias de ellas, es más si no se define un dominio concreto podemos utilizar esa propiedad para cualquier clase que queramos.

Otra consecuencia de cómo se declaran las propiedades en RDF, es que como son propiedades declaradas globalmente no se le puede asignar un rango en función de a qué clase pertenece. Por ejemplo si tenemos una propiedad ex:hasParent querríamos que si estamos hablando de una clase ex:Human su valor sea ex:Human mientras que si estamos hablando de la clase ex:Tiger nos gustaría que su valor fuera del tipo ex:Tiger. Esta diferenciación no es posible. Esta diferenciación se podrá hacer con otros esquemas que serán discutidos más adelante.

En RDF la declaración de instancias de clases no es tan rígida como puede ser en otros lenguajes. Es decir, en java por ejemplo si hacemos una clase Libro con una propiedad autor cuando declaramos un Libro tiene que tener necesariamente la propiedad autor ya sea vacía o no pero la propiedad va a estar siempre, y ninguna propiedad más si no ha sido declarada dentro de la clase. En RDF las cosas son más flexibles, el hecho de decir que una propiedad se aplica a una clase no quiere decir que tenga que existir necesariamente, es más, puede tener más propiedades que las que se le asignan directamente.

e. Otra información sobre los esquemas

Los esquemas RDF ofrecen también una serie de propiedades que pueden ser utilizadas para documentar o dar información sobre un esquema RDF o sobre instancias. Por ejemplo las siguientes propiedades:

- rdfs:comment: Añadir un comentario en lenguaje natural.
- rdfs:label: más información sobre el nombre del recurso.

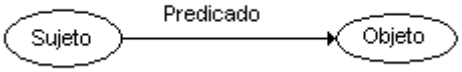
- rdfs:seeAlso: para añadir referencias a otras fuentes que toquen temas relacionados.
- rdfs:isDefinedBy: es una subpropiedad de la anterior.

(Referencia: http://www.w3.org/TR/rdf-primer/)

1.9 - RDF vs XML

1.9.1 - Introducción

Como ya hemos dicho el modelo de RDF está formado por ternas (sujeto, objeto y predicado) que pueden ser implementadas, interpretadas y almacenadas con facilidad. La representación de estas ternas serían grafos dirigidos del tipo:



En e-aula se utiliza RDF para expresar metadatos. Los metadatos son información que se asocia al recurso para facilitar su localización y procesamiento automático. RDF permite incluir propiedades y valores pertenecientes a un recurso (URL) como meta información del documento. Por lo que permite incluir metadatos creados por otras personas u organizaciones.

Además, el modelo de RDF permite definir clases como cualquier lenguaje de programación y con ellas generar una jerarquía a través de la cual se podrá inferir cierta información.

El modelo de RDF es modular, un documento RDF es el resultado de la unión de varios de esos grafos con los que se representa una sentencia RDF, de los que se concluye que:

- La información puede ser procesada de forma paralela
- En presencia de una información parcial la salida puede continuar siendo un modelo de RDF consistente que puede ser procesado con éxito.

Gracias a los namespaces de XML, una descripción puede estar almacenada en un documento XML de forma completamente transparente. Esto facilita enormemente el mantenimiento de ambos.

1.9.2 - Ventajas de RDF sobre XML

El modelo de RDF ofrece una solución flexible y atractiva para cualquier diseñador de páginas web y concretamente para un diseñador de cursos de e-learning que quiera seguir un estándar para clasificar los OE y hacerlos accesibles, tanto para una persona como para una herramienta.

Pero veamos concretamente que ventajas tiene el RDF frente al XML.

- RDF está basado en el XML de forma que cualquier documento XML puede ser convertido a RDF sin mucha dificultad. De hecho el RDF es equivalente al XML mas una serie de restricciones estructurales. Por ello ofrece una potencia mayor que el XML.

- La principal ventaja es su potencia y flexibilidad, ofrece muchas posibilidades a la hora de asociar significados a un esquema realizado con RDF. Permite expresar semántica de tal manera que es posible interpretarlo de forma automática.
El modelo RDF lleva implícito un significado que no lleva el XML. Es decir, si parseamos un documento XML la máquina no será capaz de saber si el elemento que está leyendo es el sujeto, la propiedad o el objeto de esa sentencia. Mientras que si parseamos un documento RDF la máquina podrá interpretar la semántica de esa sentencia pudiendo distinguir entre lo que es el sujeto, la propiedad o el objeto.

Ejemplo:
La sentencia “el autor de El Señor de los Anillos es Tolkien” se puede representar en XML de múltiples maneras, algunas de ellas son:

```
1.-
<autor>
<libro>El Señor de los Anillos</libro>
<autor>Tolkien</autor>
</autor>

2.-
<documento>
<detalles>
        <libro>El Señor de los Anillos</libro>
        <autor>
                <nombre>Tolkien</nombre>
        </autor>
</detalles>
</documento>
```

Si pasamos esto a un parseador que no entiende el significado de las palabras autor, detalles, libro...etc no podrá sacar ningún significado en concreto. Ya que no sabrá qué relación hay entre autor y libro, cual es el predicado, cual el sujeto...

Veamos este mismo ejemplo en RDF, una forma de representarlo sería:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:s="http://www.w3.org/bibliografia#">

<Description about="El Señor de los Anillos" s:autor="Tolkien"/>

</rdf:RDF>
```

Si pasamos este modulo de RDF a un parseador, éste será capaz de decirnos que el sujeto es “Tolkien”, que el valor es “El Señor de los Anillos” y que el predicado es “autor”. Con lo cual ya tenemos un significado implícito que puede interpretar una herramienta.

- El orden en un documento RDF no importa. Independientemente de su orden, tiene una interpretación semántica propia que se puede representar mediante un grafo dirigido como veremos en los ejemplos. Esto permite estructurar la información de manera que podemos establecer relaciones complejas entre entidades.

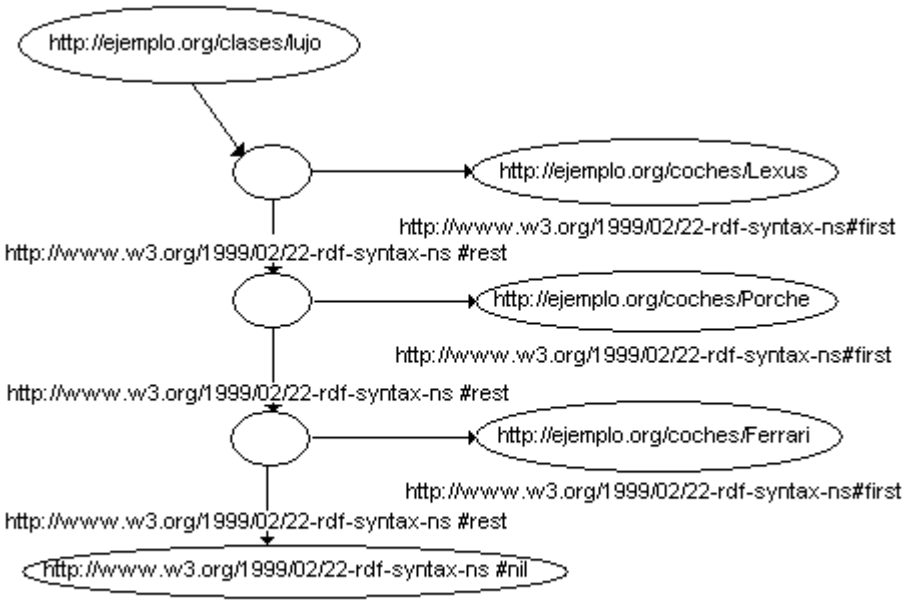
- La única semántica que ofrece XML es una estructura de árbol, no permite la expresión de relaciones más complejas que las puramente jerárquicas. Mientras

que RDF tiene la semántica de un sistema orientado a objetos y por lo tanto se puede ver como objetos que tienen propiedades y relacionarlos con otros objetos.

- Otra de las ventajas del RDF es la descentralización de contenidos, podemos usar estructuras presentes en diferentes ficheros, asociando identificadores que los relacionan. De esta manera se evita repetir aquellas estructuras que se usan en varios puntos. Esto en XML es diferente porque los elementos son autocontenidos, es decir, todo el significado que queremos que contenga un elemento debe tenerlo en el mismo sitio donde lo definimos. XML en comparación con RDF es mucho más monolítico.

- Además, RDF permite inferir información. Veámoslo con ejemplos:

  - Ejemplo1: Veamos el ejemplo de una lista implementada en RDF. El siguiente gráfico muestra una lista que representaría la sentencia "Los coches que la clase de lujo son Lexus, Porche y Ferrari"



En este gráfico, cada miembro de la colección es el valor de una propiedad rdf:first cuyo recurso es una fuente (o un nodo en blanco como en este caso) que representa una lista. Esta lista está unida al resto de listas con la propiedad rdf:rest. El final de la lista se señala con la propiedad rdf:rest señalando a rdf:nil.
El lenguaje RDF define las propiedades rdf:first y rdf:rest como del tipo rdf:List así que la información del hecho de que estos nodos son listas puede ser inferida y no es necesario declararlo con un rdf:type en cada nodo.

```
<xml version="1.0"?>
 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:c="http://ejemplo.org/coches/vocab#">
   <rdf:Description rdf:about="http://ejemplo.org/clases/lujo">
     <c:coches rdf:parseType="Collection">
         <rdf:Description rdf:about="http://ejemplo.org/coches/Lexus"/>
         <rdf:Description rdf:about="http://ejemplo.org/coches/Porche"/>
         <rdf:Description rdf:about="http://ejemplo.org/coches/Ferrari"/>
     </c:coches>
   </rdf:Description>
 </rdf:RDF>
```

- Ejemplo2: Veamos como actúa la inferencia en la definición de clases. Definamos las clases y las subclases con ternas para que resulte más sencillo .

  Definimos una clase vehículo a motor como sigue:

  ej:VehiculoMotor rdf:type rdfs:Class

  Definimos las clases de los tipos de vehículos:

  ej:Monovolumen rdf:type rdfs:Class
  ej:Camion rdf:type rdfs:Class

  Definimos también las clases de los tipos de monovolumenes:

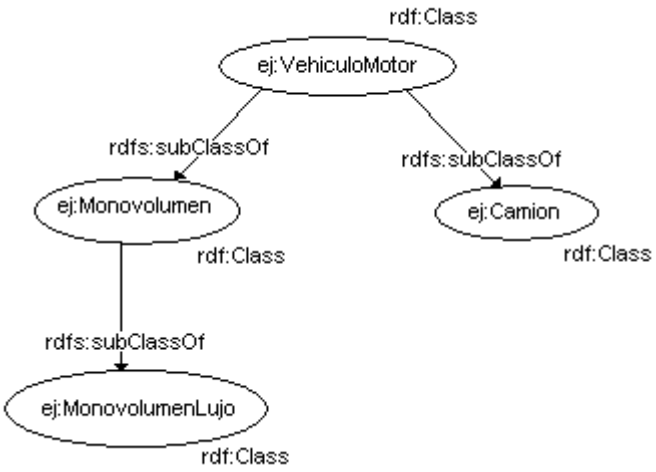
  ej:MonovolumenLujo rdfs:subClassOf ej:VehiculoMotor

  ¿Cómo definimos que un monovolumen es subclase de vehículo a motor?

  ej:Monovolumen rdfs:subClassOf ej:VehículoMotor

  El significado de rdfs:subClassOf es que cualquier instancia de ej:Monovolumen es también una instancia de la clase ej:VehículoMotor. Por lo tanto si la fuente ej:MonovolumenLujo es una instancia de ej:Monovolumen entonces, el modelo RDF permite inferir que ej:MonovolumenLujo también una instancia de ej:Vehiculomotor.

  Por lo tanto se podría inferir, el siguiente gráfico:



- Cuando representamos un esquema LOM con XML cada elemento viene representado de una misma manera. En RDF, la semántica de cada elemento LOM decide su representación. Por ejemplo:

Si hablamos de una propiedad que está almacenada en una url podemos utilizar el resource.

Si es un recurso que tiene unas determinadas propiedades podemos utilizar un objeto con una propiedad determinada. O si es solo una de las categorías que no tiene un objeto o una semántica concreta se pueden utilizar los namespaces.

Uno de los peligros que tiene el binding de los esquemas con RDF o XML es que se pierda significado, este peligro se reduce con el RDF. Se pueden obtener estructuras mucho más ricas a todos los niveles en RDF que en XML.

2 - Descripción Funcional de la Herramienta: un editor y creador de metadatos

2.1 – Existencia de distintos estándares y tecnologías. Búsqueda de la interoperbilidad. Esquemas utilizados LOM, Dublín Core y e-aula.

2.1.1 - Introducción teórica de los estándares utilizados (LOM, DC y e-aula).

En la herramienta se utilizan los estándares de metadatos Dublin Core (DC), LOM y uno propio de e-aula basado en LOM.

DC es un estándar creado en 1995, que define un sistema básico de metadatos utilizados para describir recursos de carácter general. Se compone de 15 elementos (con el significado implícito en sus nombres): title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, rights.

En Junio del 2002, la IEEE aprueba la primera versión del estándar LOM. LOM se convierte de manera gradual en el estándar de referencia para sistemas que manejan objetos educacionales de variadas clases.

El modelo de metadatos del estándar LOM describe recursos educativos basándose en un conjunto de más de 70 atributos, agrupados en nueve categorías:

- General: Agrupa la información general que describe al LO en su totalidad.
- Lifecycle: Describe la historia y el estado actual del LO y aquellas entidades que han afectado al LO durante su evolución.
- Meta-metadata: Describe el registro de metadatos asociado al LO. Cómo la instancia de metadatos es identificada, quién creó esa instancia, como, cuando y con que referencias.
- Technical: Describe los requerimientos técnicos y las características del LO.
- Educational: Describe la clave educativa o las características pedagógicas del LO.
- Rigths: Describe los derechos de propiedad intelectual y las condiciones de uso del LO.
- Relation: Define las relaciones entre este y otros LO's.

- Annotation: Proporciona comentarios del uso educativo del LO, e información sobre cuando y quién creó los comentarios.
- Classification: Describe si este LO cae en un sistema de clasificación particular.

Cada una de ellas posee diferentes atributos que les dan significado. La totalidad de los mismos se pueden consultar en los esquemas implementados o bien en la herramienta. Se puede observar un extracto de los atributos que contiene en la selección de los mismos que realiza el esquema de e-aula.

El esquema de e-aula, basado en LOM, posee los siguientes atributos divididos por categorías:

1.General
- 1.1 identifier
  - 1.1.1 catalog
  - 1.1.2 entry
- 1.2 title
- 1.3 language
- 1.4 description
- 1.5 keyword

2.Lifecycle
- 2.3 contribute
  - 2.3.1 role
  - 2.3.2 entity
  - 2.3.3 date

3.Meta-metadata
- 3.2 contribute
  - 3.2.1 role
  - 3.2.2 entity
  - 3.2.3 date
- 3.3 Metadata Schema
- 3.4 Language

4.Technical
- 4.1 format
- 4.2 size
- 4.3 location

5.Educational
- 5.1 interactivity type
- 5.2 learning resource type
- 5.5 intended end user role

9.Classification

Las categorías Lifecycle, Annotation y Relation tienen un sentido claro de mantenimiento del LO, por ello se pueden asociar a los LO's diferentes juegos de atributos de estas categorías.

La categoría Classification tiene un tratamiento especial dentro de los esquemas que manejamos en la herramienta. La naturaleza de esta categoría es diferente a la del resto del estándar. Mientras que en el resto se rellenan metadatos que se

asocian al LO, en este apartado debemos disponer de clasificaciones establecidas previamente y estas son las que nos ofrecen los valores que podemos asociar al LO. Todo lo que tiene que ver con las taxonomías; su formato, su implementación concreta del estándar LOM y el uso que hacemos de las mismas en la herramienta,… será analizado en el siguiente punto de la memoria.
Uno de los fines específicos de la herramienta es facilitar la labor de rellenar los valores de metadatos a los creadores de cursos, y en concreto, por medio del esquema e-aula a los responsables del proyecto e-aula. Por ello, este esquema está adaptado a los contenidos necesarios de este proyecto; cómo hemos visto, es una selección de los atributos de LOM que se han considerado dentro de este ámbito, la utilidad en este caso compromete la interoperabilidad.
Pero por otro lado, la herramienta ofrece una utilidad global de clasificación de LO's atendiendo a los esquemas generalizados LOM y DC lo que la dota de universalidad, favoreciendo la interacción de los contenidos con otras aplicaciones.

2.1.2 - Cómo se representan y utilizan estos esquemas en nuestra herramienta.

Estos estándares son el núcleo de nuestra colección de metadatos. Nuestra aplicación asocia metadatos a los LO's de acuerdo a estos esquemas y almacena en ficheros esta información.

En este punto es dónde entra RDF, usamos esta tecnología para implementar los esquemas de metadatos que acabamos de ver. Los ficheros que contienen la información de los metadatos asociados a los LO's (resultados del proceso de la herramienta) también estarán codificados con RDF, mediante esta tecnología podemos relacionar fácilmente ambos tipos de fichero.

El formato de estos ficheros y las estructuras de RDF seleccionadas para crearlos se analiza en el siguiente epígrafe. En el caso de DC utilizamos directamente el esquema oficial implementado en RDF, pero en el caso de LOM y e-aula hemos desarrollado una adaptación del estándar original, lo que se denomina binding, ya que no existe ninguna implementación oficial en RDF. Este binding ha sido realizado estudiando las posibilidades que ofrecía la tecnología RDF y analizando diferentes ejemplos encontrados en la red.

2.1.3 - Características de nuestro binding de LOM.

Cómo hemos visto, los atributos del esquema LOM están contenidos en una estructura de árbol debajo de sus categorías, este árbol hace posible organizar la información de una forma consistente, agrupándola en porciones relacionadas. De esta forma podemos decir que el modelo de metadatos de LOM es contenedor recursivo.
Por otro lado, en el conjunto de elementos de metadatos Dublin Core, los recursos vienen definidos por propiedades tales como título, creador, materia, descripción, tipo, fecha… es por tanto, un modelo de metadatos basado en la relación atributo-valor.

Los metamodelos LOM y DC son claramente incompatibles, como ejemplo; las relaciones de los atributos de LOM pueden ser más complejas que en DC, podemos tener atributos que no hacen referencia al LO en sí, sino al propio conjunto de

metadatos, o a la persona que ha contribuido, mientras que en DC van asociados al objeto que clasifican.
Analizando esta diferencia de modelos y la compatibilidad de la tecnología RDF con el modelo de datos de DC (también es no-contenedor y basado en relaciones propiedad-valor) se puede afirmar que no es sencillo establecer una representación directa de los atributos de LOM con las estructuras RDF, por el contrario, en el caso de DC la representación se basa en una traducción de las estructuras.
Pero cómo hemos visto en otros puntos de esta memoria, RDF dispone de un completo conjunto de sentencias para construir este binding. A continuación se exponen las características más notables de las elegidas para nuestra implementación.

Disponemos de un fichero por cada una de las categorías de LOM, en cada uno de ellos se presentan todos los atributos de las mismas.

Dentro de nuestros ficheros a menudo invocamos a estructuras presentes en otros ficheros (como las sentencias propias de rdf, o bien vocabularios concretos…). La forma de hacerlo es la siguiente:

Normalmente al comienzo, acompañando a la etiqueta de apertura del fichero RDF se codifican las entradas que asocian un prefijo con una URL, el contenido de la URL tiene las estructuras que podemos invocar, y la forma de hacerlo es usando el prefijo definido.

```
<rdf:RDF xml:lang="en" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
                xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
```

Dentro del contenido de la URL apuntada por “rdf” tenemos por ejemplo la definición de la estructura Property de RDF.

```
…
<rdfs:Class rdf:about="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property">
  <rdfs:isDefinedBy rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
  <rdfs:label>Property</rdfs:label>
  <rdfs:comment>The class of RDF properties.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdfs:Class>
…
```

Para utilizar esta estructura dentro del código RDF de nuestro fichero haremos esta llamada rdf:Property.

Este mismo proceso se utiliza para relacionar los ficheros de metadatos resultantes de la clasificación realizada por nuestra herramienta, con los ficheros dónde residen los esquemas de atributos. La función en ese caso es asociar en este fichero resultado los valores de los metadatos con su atributo correspondiente, situado en los diferentes ficheros de esquemas.

Este concepto queda más claro según avanzamos por la explicación. Analizamos a continuación las estructuras de los ficheros que implementan los esquemas.

Todos los elementos de estos ficheros han sido construidos como propiedades (rdf:Property) y clases (rdf:Class). Las propiedades representan los diferentes atributos de una categoría y las clases definen los valores predefinidos de aquellos atributos que los soportan.

Algunos atributos de ciertas categorías tienen atributos predefinidos o aconsejados por LOM, se trata de valores fijos. En la práctica, en los formularios se presentan en forma de select de tal forma que no nos podemos salir de lo establecido. De cara a los esquemas también conlleva una construcción especial.
Ej: “role” dentro de “Meta-metadata” tiene los valores: “author” y “validator”.

Veamos un fragmento del fichero dónde se muestra la construcción utilizada para representar los atributos. Estamos ante “version” de la categoría “lifecycle”.

```
<rdf:Property rdf:ID="version">
 <rdfs:label>Version</rdfs:label>
 <rdfs:domain rdf:resource="&ruta;LO"/>
 <rdfs:range rdf:resource="&xsd;string"/>
 <rdfs:comment>2.1: La edicion de este LO.</rdfs:comment>
</rdf:Property>
```

Analicemos a continuación esta estructura, generalizando sus contenidos.
- Todas las propiedades y clases vienen identificadas por rdf:ID, se trata de un identificador de uso interno en los esquemas (coincide con los nombres de los atributos en LOM). Los atributos que a su vez se componen de otros vendrán identificados por la construcción rdf:nodeID, los analizaremos en detalle más adelante.

- El cuerpo de las propiedades está formado por unas cuantas sentencias rdf que pasamos a comentar:

- rdfs:label. Etiqueta o nombre de los atributos según LOM
- rdfs:comment. Comentario o explicación del atributo, lleva asociado el número asociado en el esquema LOM
- rdfs:domain. Dominio del atributo, entidad a la que se aplica la propiedad. El dominio de la mayoría de las propiedades es un LO, y apunta a un fichero RDF dónde hemos representado un LO genérico. En los atributos de metametadata no existe esta etiqueta, puesto que el dominio de los mismos es el mismo fichero de metadatos serializado.
- rdfs:range. Tipo del valor del atributo. Si es un atributo de valores predefinidos hará referencia a una clase presente en el mismo documento, si son atributos simples como string, date, language… hacen referencia al esquema estándar de xml (http://www.w3.org).

Algunas de ellas se hacen extensivas a otras estructuras utilizadas, por ejemplo en rdf:class.

El enlace de las sentencias domain y range con las URL’s es propio de xml, viene representado por medio de rdf:resource y el identificador que llama a estas URLs.

```
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
                   <!ENTITY ruta "http://www.eaula.es/RDF/generico#">]>
```

Algunos casos de atributos especiales:
- Los atributos que están compuestos por otros atributos tienen esta forma:

```
<rdf:Property rdf:nodeID="identifier">
 <rdfs:label>Identifier</rdfs:label>
 <rdfs:comment>1.1: Una etiqueta unica y global que identifique al
LO</rdfs:comment>
</rdf:Property>

<rdf:Description rdf:nodeID="identifier">

 <rdf:Property rdf:ID="catalog">
  <rdfs:label>Catalog</rdfs:label>
  <rdfs:domain rdf:resource="&ruta;LO"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:comment>1.1.1: El nombre o disenyador del esquema de catalogacion de
este LO</rdfs:comment>
 </rdf:Property>

 <rdf:Property rdf:ID="entry">
  <rdfs:label>Entry</rdfs:label>
  <rdfs:domain rdf:resource="&ruta;LO"/>
  <rdfs:range rdf:resource="&xsd;string"/>
  <rdfs:comment>1.1.2: El valor del identificador del esquema de catalogación de
este LO</rdfs:comment>
 </rdf:Property>

</rdf:Description>
```

Primero se presenta la propiedad de más alto nivel y luego dentro de la construcción rdf:Description, se presentan los atributos del siguiente nivel con el formato ya visto. A los casos de más de dos niveles, se extiende el mismo convenio.

- Para los atributos con valores predefinidos, recomendados por el estándar LOM, usamos la siguiente construcción:

```
<rdfs:Class rdf:ID="Structure">
 <rdfs:comment>Instancias de esta clase representan tipos de
Structure</rdfs:comment>
</rdfs:Class>
<Structure rdf:ID="atomic"/>
<Structure rdf:ID="collection"/>
<Structure rdf:ID="networked"/>
<Structure rdf:ID="hierarchical"/>
<Structure rdf:ID="linear"/>

<rdf:Property rdf:ID="structure">
 <rdfs:label>Structure</rdfs:label>
 <rdfs:domain rdf:resource="&ruta;LO"/>
 <rdfs:range rdf:resource="#Structure" />
 <rdfs:comment>1.7: Organizacion estructural de este LO.</rdfs:comment>
</rdf:Property>
```

Primero se presenta la clase, utilizando el mismo id que para la propiedad (primera en mayúscula, para diferenciarlas), y se acompaña de los posibles valores. A continuación definimos el atributo, con la etiqueta de rango igual a la clase anterior. Cuando hacemos referencia a un identificador presente en un documento RDF dentro de una sentencia rdf:resource, utilizamos la construcción “url#id”. Obviando la url si se trata de un identificador presente en el mismo documento, como en este caso. Esta misma construcción se utiliza también en el fichero resultado de metadatos.

El fichero resultante de la clasificación que realiza la herramienta tiene las siguientes características:

- Primero presenta los prefijos para las diferentes categorías de LOM, igual que vimos anteriormente:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:lom-gen="http://localhost/LOM-General#"
      xmlns:lom-ann="http://localhost/LOM-Annotation#"
      xmlns:lom-lif="http://localhost/LOM-Lifecycle#"
      xmlns:lom-cla="http://localhost/LOM-Classification#"
      xmlns:lom-edu="http://localhost/LOM-Educational#"
      xmlns:lom-met="http://localhost/LOM-Metametadata#"
      xmlns:lom-rel="http://localhost/LOM-Relation#"
      xmlns:lom-rig="http://localhost/LOM-Rights#"
      xmlns:lom-tec="http://localhost/LOM-Technical#">
```

- A continuación, como cuerpo de las etiquetas rdf:Description se muestran los atributos de las diferentes categorías.

```
<rdf:Description about="nombre_de_LO">
...
...
</rdf:Description>
```

Siguiendo el convenio de invocar a estructuras presentes en otros ficheros, a los que se llega por medio de los prefijos antes definidos, este es el aspecto que presentan las líneas RDF que asocian los valores introducidos por formulario a los diferentes atributos:

```
<lom-rel:description>Relación básica para la programación</lom-rel:description>
```

Si el atributo es de valores predefinidos, lo presentaremos mediante la construcción comentada anteriormente:

```
<lom-rel:kind resource="http://localhost/LOM-Relation#haspart"/>
```

Cuando accedemos a atributos de atributos lo hacemos mediante un formato de puntos. Como ejemplo; de la categoría “Relation”, “catalog” es un atributo de “identifier” que lo es de “resource”.

```
<lom-rel:resource.identifier.catalog>ISBN</lom-rel:resource.identifier.catalog>
```

Cuando tenemos más de un juego de atributos se presentan de la misma manera los diferentes valores, y llevan antes un comentario que especifica a que instancia del juego estamos haciendo referencia.

De nuestro binding podemos extaer algunos rasgos fundamentales que lo caracterizan:
- Estructura. La división de las categorías y sus atributos en diferentes ficheros hace que la estructura sea portable, manejable y reutilizable.
- Representación de datos. Cada atributo está representado de acuerdo a su significado, existen valores predeterminados por LOM y valores simples como cadenas de caracteres, enteros, fechas...
- Seguridad de tipos: Los atributos tienen determinado el rango de sus valores y el dominio dónde son aplicados, de esta forma se gestionan los valores posibles.
- Simplicidad técnica: Las estructuras RDF y los convenios elegidos hacen que sea fácil de mantener, intuitivo y por lo tanto entendible.

2.2 – Taxonomías ¿qué son? ¿Qué utilidades tienen en e-learning? ¿Para que las usamos?

2.2.1 - Descripción, ¿qué son?

Las taxonomías son una forma de clasificación. Se define como la ciencia que se encarga de los principios, métodos y fines de la clasificación.

Es un conjunto de información que representa la organización jerárquica de un ámbito del conocimiento siguiendo un protocolo de género-especie o general-específico. Gracias a la taxonomía se organiza la estructura y se guía el indexado, etiquetado y asociaciones de contenidos.

La forma más habitual de una taxonomía es una jerarquía. En el nivel más alto, se usan términos generales o frases descriptivas. Cada uno de los términos generales puede ser jerarquizado en un grupo de términos que proporcionan más refinamiento de los anteriores. Cada uno de estos términos del segundo nivel puede tener por debajo un conjunto de términos que lo refinanm,… formando una clasificación completa.

Como ejemplo, los términos del nivel superior del Esquema de Clasificación de la Biblioteca del Congreso de los EEUU (LCC) son:

- A -- GENERAL WORKS
- B -- PHILOSOPHY. PSYCHOLOGY. RELIGION
- C -- AUXILIARY SCIENCES OF HISTORY
- D -- HISTORY: GENERAL AND OLD WORLD
- E -- HISTORY: AMERICA
- F -- HISTORY: AMERICA
- G -- GEOGRAPHY. ANTHROPOLOGY. RECREATION
- H -- SOCIAL SCIENCES
- J -- POLITICAL SCIENCE
- ...

La categoría "B -- PHILOSOPHY. PSYCHOLOGY. RELIGION" tiene un grupo extenso de subcategorías. Una de ellas es F, Psychology. Psychology que a su vez, también está dividida, e incluye 180-198.7, Experimental psychology.

La descomposición de taxonomías se corresponde con un determinado punto de vista. Por ejemplo, el término *humano* puede descomponerse en subtipos dependiendo de factores físicos (hombre negro, hombre blanco) o sociales (político, médico, abogado), entre otros muchos. Como consecuencia, una determinada palabra puede tener cabida en distintos nodos de la taxonomía.

Algunos ejemplos reales, pueden darnos una idea global de la utilidad de las taxonomías.

- La comentada anteriormente LCC es una taxonomía de materias mantenida para la biblioteca del congreso de USA. Muestra perfectamente la utilidad de las taxonomías con uno de los ejemplos a los que se hace referencia en multitud de ocasiones cuando se habla de taxonomía; materias de una biblioteca.

http://lcweb.loc.gov/catdir/cpso/lcco/lcco.html

- "La Clasificación de Materias Matemáticas 2000 (MSC)" se utiliza para categorizar elementos cubiertos por las dos bases de datos de referencia matemática, Mathematical Reviews (MR) y Zentralblatt MATH (Zbl). EL MSC se compone de alrededor de 5,000 elementos, cada uno de ellos corresponde a una disciplina matemática (Por ejemplo: 11 = Number theory; 11B = Sequences and sets; 11B05 = Density, gaps, topology)

http://www.ams.org/msc/

- El "Listado de Materias Médicas" (MeSH), comprende el vocabulario controlado de la Biblioteca Nacional de Medicina (NLM) de USA. Consistente en términos organizados en una estructura jerárquica que permite buscar en varios niveles de la especificación.
Esta taxonomía es usada por la NLM para indexar artículos, catalogar libros, documentos o material audiovisual adquirido por la biblioteca. Cada referencia bibliográfica está asociada con un conjunto de términos MeSH que describen su contenido. De manera similar, se utiliza el vocabulario MeSH para buscar elementos asociados con un asunto determinado.

http://www.nlm.nih.gov/mesh/filelist.html

- XBRL (*extensible Business Reporting Language)* nace con el fin de simplificar la automatización del intercambio de información financiera mediante el uso de XML, utilizan el concepto de taxonomía como indicador de las líneas maestras sobre las que se fundamenta el intercambio de información, lo que hace que el tratamiento de los datos se simplifique enormemente.
Las taxonomías son, por lo tanto, los diccionarios del lenguaje XBRL. Consisten en esquemas de clasificación que definen etiquetas específicas para cada elemento de específico de información (por ejemplo, "Beneficio Neto").

http://www.xbrl.org.es/index.html

Estos son algunos ejemplos que nos muestran la función principal de las taxonomías; la clasificación de información.

2.2.2 - Utilidad de las taxonomías en e-learning.

En los ejemplos analizados anteriormente se observan clasificaciones de la información manejada en determinados entornos; así teníamos una taxonomía de las materias de una biblioteca, o de términos de medicina en un entorno médico… Dentro de e-learning sucede algo similar, el fundamento de la información manejada en este tipo de aprendizaje son los learning objects y estos pueden ser igualmente clasificados atendiendo a diferentes razones o características.

La discusión de las características básicas de los LO's, tales como la secuencia, alcance o la estructura, nos lleva a considerar que pueden existir diferentes tipos de LO. Y por consiguiente se hace necesario un sistema de clasificación, esta necesidad se hace aún más evidente cuando se establece un ámbito de los LO's y pueden ser agrupados o relacionados dentro del mismo. Las taxonomías han acompañado históricamente al diseño de teorías educacionales, aunque no existe una taxonomía general de clasificación de LO's, esta falta se palia con el desarrollo de taxonomías especificas por parte de las aplicaciones.

Algunos ejemplos de taxonomías reales relacionadas con el aprendizaje:

- Taxonomía que clasifica el dominio del aprendizaje (Bloom, 1956). Los valores de este dominio son cognitive, affective, psichomotor que se dividen a su vez en más subcategorías y que hacen referencia al dominio de inteligencia, actitud y habilidad respectivamente.

- Taxonomía de LO's con respecto a su uso educacional (Wiley,2001), se distinguen cinco categorías: fundamental, combined-closed, combined-open, generative-presentation, generative-instructional. Se establecen con ellas diferentes tipos de LO's disponibles, ejemplo de fundamental es un JPEG de una mano tocando un acorde de piano, y de generative-presentation es un applet de JAVA capaz de gráficamente generar un conjunto elementos musicales, y luego posicionarlos apropiadamente para presentar un problema de identificación de un acorde a un estudiante.

Resumiendo, existen algunas taxonomías generadas especialmente para e-learning, pero podemos asociar a los LO's clasificaciones de muy diverso origen en concordancia con nuestra aplicación u objetivo.

2.2.3 - Taxonomías y nuestra aplicación

a. Situación general

Cómo se ha comentado en el punto anterior, dos de nuestros esquemas están basados en el estándar de metadatos de objetos educativos LOM. Por un lado tenemos el esquema LOM puro y por otro la adaptación de LOM a las necesidades de e-aula. Dentro de estos esquemas es dónde se sitúan las taxonomías que utilizamos en nuestra aplicación, concretamente dentro de la categoría Classification.

Según el estándar LOM esta categoría describe si el LO cae en un sistema particular de clasificación. Para establecer los valores de esta clasificación, la categoría establece los siguientes atributos:

- *Purpose*: Propósito de la clasificación.
- *Taxon Path*: La clasificación en sí, compuesta por nodos taxon.
- *Source*: Fuente de la clasificación.
- *Taxon*: Estructura básica de la clasificación.
  - *Id*: Identificador del taxon.
  - *Entry*: Label del taxon.
- *Description*: Descripción de la taxonomía.
- *Keyword*: Palabras clave de la taxonomía.

Estos atributos son respetados por las taxonomías que forman parte de nuestra aplicación.

b. Esquemas RDF para taxonomías.

Las taxonomías son almacenadas en ficheros RDF de acuerdo a un esquema establecido, al igual que sucedía con el resto de categorías de LOM (binding de LOM).

De esta forma las taxonomías son registradas en dos archivos.
Por un lado en el archivo que representa la categoría Classification de LOM, dónde se establece una entrada a la taxonomía.
Y por otro lado en el fichero RDF de la taxonomía concreta, dónde se hace referencia al anterior y dónde se almacenan los valores de la taxonomía analizados anteriormente.
A continuación se presentan los fragmentos fundamentales de dichos ficheros.

Fichero Classification de LOM. (LOM-Classification.rdf).
En él se muestra la entrada de nuestra taxonomía como una propiedad, que a su vez es subpropiedad de classification, esta entrada se usará en el serializado de metadatos. Y además dos clases Taxonomy y taxon que van a ser utilizadas desde el fichero de la taxonomía analizado a continuación.

```
<?xml version="1.0"?>
...

<rdf:Property rdf:ID="classification">
   <rdfs:label>Classification</rdfs:label>
   <rdfs:comment>Esta categoria describe si este LO cae en un sistema particular
de clasificacion.</rdfs:comment>
</rdf:Property>

<rdf:Property rdf:ID="Informatica">
    <rdfs:subPropertyOf rdf:resource="#classification"/>
    <rdfs:label>Especialidades Informática</rdfs:label>
</rdf:Property>

<rdfs:Class rdf:ID="Taxonomy">
    <rdfs:label>Taxonomy</rdfs:label>
    <rdfs:comment>Una taxonomia jerarquica.</rdfs:comment>
</rdfs:Class>

<rdf:Property rdf:ID="taxon">
    <rdfs:label>Taxon</rdfs:label>
```

```
   <rdfs:comment>Esta propiedad describe un termino particular de una
taxonomia. Un taxon es un nodo que tiene un label definido o
termino.</rdfs:comment>
  </rdf:Property>
</rdf:RDF>
```

Fichero de la taxonomía. (LOM-tax_Informatica.rdf)

Haciendo uso de las clases del fichero anterior va montando los diferentes niveles del árbol taxonómico, compuestos por el id y entry comentado anteriormente.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcq="http://dublincore.org/2000/03/13/dcq#"
xmlns:lom-cla="http://localhost/LOM-Classification#"
xmlns:systax="http://localhost/LOM-tax_Informatica#">

<lom-cla:Taxonomy rdf:ID="INFORM">
 <rdfs:label>Clasificar las distintas especialidades de informática</rdfs:label>

 <lom-cla:taxon>
  <systax:INFORM rdf:ID="I.G">
   <rdf:value>I.G</rdf:value>
   <rdfs:label>Gestion</rdfs:label>
  </systax:INFORM>
 </lom-cla:taxon>

 <lom-cla:taxon>
  <systax:INFORM rdf:ID="I.S">
   <rdf:value>I.S</rdf:value>
   <rdfs:label>Sistemas</rdfs:label>

   <lom-cla:taxon>
    <systax:INFORM rdf:ID="I.S.R">
     <rdf:value>I.S.R</rdf:value>
     <rdfs:label>Redes</rdfs:label>
    </systax:INFORM>
   </lom-cla:taxon>

   <lom-cla:taxon>
    <systax:INFORM rdf:ID="I.S.P">
     <rdf:value>I.S.P</rdf:value>
     <rdfs:label>Programacion</rdfs:label>
    </systax:INFORM>
   </lom-cla:taxon>

  </systax:INFORM>
 </lom-cla:taxon>

</lom-cla:Taxonomy>
```

Fichero serializado de metadatos (nombrefichero.rdf)
La entrada serializada del fichero de metadatos del LO correspondiente usando esta taxonomía tiene este aspecto:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
...
xmlns:lom-cla=http://localhost/LOM-Classification#
...

<lom-cla:Informatica resource="http://localhost/LOM-tax_Informatica#I.S.R" />
...
</rdf:RDF>
```

c. Situación específica.

En este punto analizamos la interacción de nuestra aplicación con las taxonomías; las clasificaciones concretas que se manejan y la posibilidad de crear nuevas.

Como hemos comentado antes, a partir de los ficheros RDF que tienen implementadas las taxonomías, la aplicación obtiene los posibles valores y se cargan en los formularios de inserción de metadatos en la categoría Classification, en forma de select. En el manual de usuario se explica este paso de manera detallada.

Dentro de la aplicación y dependiendo del esquema, manejamos una filosofía diferente de taxonomías.
En el esquema de metadatos de LOM adaptado a e-aula disponemos de una taxonomía del dominio de los lenguajes de programación creada de manera estática. Los posibles valores que se podrán añadir a los LO's seleccionados con este esquema se corresponderán con los tipos de lenguajes de programación.

Pero si seleccionamos LOM, podemos crear una nueva taxonomía siguiendo el estándar y formará parte de las taxonomías de la aplicación. Se nos mostrará la lista de las disponibles y seleccionando podremos clasificar nuestro LO con sus valores. Igualmente este proceso se detalla en el manual de usuario.

2.3 - Análisis crítico y comparativo de otras herramientas.

Buscando en el mercado, por Internet, podemos encontrar varias herramientas que hacen una función parecida a la nuestra.
Cada una de estas herramientas están enfocadas para distintos entornos o funcionalidades.

Antes de empezar a realizar la codificación de nuestra herramienta, durante el proceso de estudio e investigación de la tecnología, hemos analizado las otras posibilidades existentes en el mercado. En ocasiones, estas nos ofrecían ayuda, y ejemplos que hemos desarrollado. En otros casos, las herramientas estaban enfocadas de manera distinta, y no nos servían.

Resulta muy difícil realizar una comparación entre aplicaciones, en ocasiones se puede caer en errores, al no tratarse de elementos que tengan una finalidad común, pero siempre es bueno fijarse en los otros ejemplos, para poder especificar

mas las características del nuestro, y situarlo en el sitio apropiado dentro del mercado.

Estas herramientas ya están analizadas individualmente en otra sección de la documentación del proyecto. Ahora vamos a pasar a analizarlas globalmente, en comparación con la nuestra.

Se pueden clasificar en dos tipos de herramientas, según sean ejecutables bajo un entorno Web, o simplemente aplicaciones que se ejecuten solas. Nuestro fin es encontrar y analizar las herramientas que se ejecuten en entornos Web, ya que la nuestra tiene ese enfoque como base principal, por los motivos ya comentados en la documentación.

Encontramos DCdot, como una herramienta Web, la cual se encarga de añadir meta información a determinadas paginas Web, especificadas por el usuario. Esto nos parece una restricción bastante importante, pues no es posible generar meta información para elementos que no sean paginas Web. La herramienta recoge ciertos datos, en caso de estar presentes en la pagina Web, como pueden ser titulo, creador y palabras claves. En caso de no estar presentes, el usuario puede añadir información. La meta información generada, se nos muestra en diversos formatos, ya sea XHTML, HTML, XML o RDF. Pero solamente puede ser codificada siguiendo el estándar de Dublin Core. Según esto, muestran la información en mucho formatos distintos, pero solamente usa un posible esquema.

Otra herramienta Web, es Reggie - The Metadata Editor.
En este caso se nos muestra como un applet, algo que requiere que la maquina virtual de java, instalada en el cliente, lo ejecute. Esto sin duda ya empieza a ser una restricción de portabilidad, sesgando la posibilidad de acceso. Hay que pensar que no todo el mundo tiene privilegios de instalación, o no quiere instalar el software adicional necesario.
La herramienta nos muestra un selector de los tipos de esquema que soporta. Y nos pide que insertemos una URL si queremos añadir meta datos a una pagina Web especifica.
Después vemos un gran formulario que nos muestra los campos correspondientes al esquema seleccionado. La mayor parte de los campos son elementos de texto, aunque según el esquema, encontramos select, que nos dejan seleccionar entre los elementos especificados sin poder añadir nada más.
Para poder obtener los meta datos, podemos seleccionar que la salida de los mismos sean en formato HTML, XHTML o RDF.
Claramente el aspecto más interesante de esta herramienta, que le da gran potencia. Es la capacidad de importar cualquier esquema, generado por nosotros mismos, especificando en que URL se encuentra. Nos ofrecen un manual on-line, para ver como debemos generar nuestros esquemas, siempre siguiendo unas normas que establecen para ser capaces de interpretarlo.

Nordic DC metadata creator. De nuevo una herramienta Web, muy parecida a la que estamos desarrollando. Esta no requiere de la instalación de software, pues todo se ejecuta en el servidor. Al cliente solo se le muestra el formulario de entrada de meta datos, y la respuesta de salida. Tenemos la opción de añadir varios valores por cada campo del esquema. Solamente implementa el esquema de Dublin Core. La codificación final de los meta datos se muestra como pagina HTML.

MetaBrowser System es una herramienta, de las que no son vía Web. Por lo tanto consideramos que no es para nada una aplicación comparable con la nuestra. No es

portable, requiere de instalación en el sistema, no es apta para todos los sistemas. Y además no es de uso libre, pues es un producto que debe ser licenciado.
Este tipo de herramientas no son comparables, y no tienen objeto de análisis en el proyecto.
Es necesario comentar que hemos encontrados otras herramientas parecidas a esta ultima, que como decimos no son objeto de nuestro estudio.

Analizar varias herramientas ha resultado una labor bastante productiva.
Hemos podido observar como el hecho de implementar varios esquemas distintos, en nuestro caso Dublin Core, LOM y LOMe (este ultimo un esquema propio, generado a partir de LOM, pero orientado al proyecto de eAula), añade un peso importante al proyecto.
A su vez, hemos visto la importancia de realizar una aplicación que se ejecute en un servidor remoto. Que en el cliente, no se ejecute ningún proceso de gran carga. Es necesario pensar que esta aplicación puede ser ejecutada simultáneamente por muchos usuarios, y para esto necesitamos una capacidad de trabajo, propia de grandes sistemas, como servidores de aplicaciones dedicados. Pero la principal ventaja, es la portabilidad, y la falta de restricciones a la hora de acceder a la utilidad.
Nosotros, para mostrar los resultados de salida, lo hacemos mediante un archivo RDF, o XML. Siguiendo las especificaciones propias de cada uno.
Otra característica muy importante, en contraposición con las herramientas analizadas, es que nosotros permitimos añadir meta datos a cualquier elemento, ya sean imágenes, paginas Web, archivos PDF etc...es lo que nosotros llamamos el Learning Object (LO), el objeto de estudio. Una vez todos los meta datos están rellenos y el usuario decide exportarlos, el resultado puede ser mostrado por pantalla en RDF o XML, pero también tiene la opción de almacenar un archivo comprimido en formato ZIP, el cual contendrá el LO, su documento RDF y el archivo XML, con la meta información generada. De esta forma, podemos tener almacenados todos los elementos a los cuales hayamos añadido meta datos. Pudiéndolos clasificar de la manera que consideremos más oportuna.
Lo único que hemos encontrado en otras herramientas, y que habría sido muy útil es la idea de incluir un módulo que fuese capaz de codificar meta datos siguiendo un esquema especifico creado por el usuario. Tal y como hacia la aplicación analizada Reggie.

2.4- Nuestra Herramienta

Muchos son los puntos fuertes, que encontramos dentro de nuestra aplicación. Tanto en cuestiones técnicas como funcionales.

Con respecto a las ventajas de la parte mas técnica vamos a hacer una breve introducción, y más adelante nos centraremos de lleno en este tema:
Encontramos una herramienta Web, con capacidad de ser usada por prácticamente el 100% de los usuarios. Esto es debido a que la parte mas pesada se ejecuta en un servidor remoto, enviándole solo al usuario texto plano en formato HTML, a modo de formularios, que solamente deben ser rellenados en la maquina cliente por el usuario. A su vez, el lenguaje en el que esta programado Java, y el tipo de método de programación seguido: MVC (Modelo Vista Controlador), nos ofrece una aplicacion completamente modularizada en clases. Está programada siguiendo los estándares de la programación orientada a objetos, con lo que no seria difícil, añadir nuevas funcionalidades en un futuro. O modificar algunas partes ya existentes para adaptarlas o mejorarlas.

Con respecto a la parte menos técnica y gracias a la tecnología implementada por debajo, vemos como es posible incluir nuevos esquemas, sin mas que programarlos como un modulo nuevo, e incluirlo en el código, no requiere mucho esfuerzo, simplemente conocer un poco la herramienta por dentro.
Dos funciones de cada uno de los módulos pertenecientes a cada esquema, nos permiten parsear los metadatos a formatos RDF o XML, de esta forma, la labor se hace por separado, pudiendo también hacer actualizaciones en el código de forma mucho más transparente, en caso de que necesitemos que la creación de los archivos XML o RDF cambia en algún punto.

Dadas las características de la herramienta, es muy fácil incluir, toda la aplicación, como una parte mas de un proyecto mucho más grande. Este es el caso de proyectos para aulas virtuales, que ha sido un poco la guía de todo este trabajo. La facilidad de esta labor, de nuevo, reside en el esquema de nuestra aplicación, y la forma en que esta codificada. Su modularidad nos facilita el proyecto de fusión con otras tecnologías o como una parte mas de un gran proyecto, siempre siguiendo la idea de que la aplicación se ejecuta vía Web. Simplemente seria necesario disponer en el servidor, de las características necesarias para ejecutar la parte más técnica. Hoy en día no es difícil encontrar un servicio que nos proporcione los requisitos que exige la herramienta.
Por esto, vemos que la herramienta en si, puede ser una parte mas, de un proyecto muy grande.

2.5 - Nuestra herramienta limitaciones.

En un primer momento, encontramos como limitaciones, el hecho de no tener tantos esquemas implementados como otras herramientas. Pero preferimos pensar que esto es solo una característica menos de nuestra herramienta, y no una limitación funcional, pues al estar tan modularizado todo en clases, es muy fácil incorporar nuevos esquemas o estándares, como tienen algunas de las herramientas analizadas.

Una limitación, más importante, la encontramos a la hora de tratar de importar nuestro esquema propio en tiempo de ejecución. Solamente hemos encontrado una herramienta, con un manual asociado que enseña como debe ser formateado un archivo RDF, como ya comentamos antes. En esta, luego el sistema puede pedirlo como entrada y es capaz de formar toda una estructura de metadatos, siguiendo ese esquema propio creado por el usuario.
En nuestro caso, seria necesario entrar a nivel de código y programar la aplicación, pudiendo así ofrecer ese nuevo esquema que el usuario demande.

Por lo demás, no vemos ningún tipo de restricción, ni limitación funcional, pues como hemos comentado en puntos anteriores, la herramienta esta implementada siguiendo unos estándares que favorecen al máximo la reutilización, adaptación y actualización del código.

3 – Documentación sobre la aplicación: diseño e implementación

3.1 - Diseño UML

El diseño UML es mostrado como un anexo a la documentación. Pues es la mejor manera de hacerse una idea de los enlaces entre las distintas clases del código. Se compone de cuatro paginas entrelazadas.

3.2 – Diagrama de funcionamiento

Como documento anexo, también incluimos los distintos diagramas de interacción de cada parte que compone nuestra herramienta. Un diagrama para la zona de creación de metadatos, otro para la zona de edición, y uno para explicar como funciona nuestro modelo de objetos que trata, codifica e interpreta los metadatos. Este anexo lo podemos encontrar como diapositivas dentro de la presentación de la aplicación.

3.3 – Especificaciones Técnicas y Tecnológicas

La aplicación, esta programada íntegramente en lenguaje Java.
La elección de este lenguaje se debe a la intención de implementar nuestra aplicación en un entorno web.
Consideramos muy importante hoy en día el mundo de internet, la amplia difusión de todos los elementos que componen la “web” es un punto a favor del proyecto.
A su vez, los navegadores web, ya se encuentran disponibles en todas las plataformas de mayor acceso, como pueden ser sistemas operativos como Windows, Machintos, Linux, Unix.
Cualquier maquina que encontramos hoy en día, ya sea a nivel de aficionado, usuario, o experto, puede soportar alguno de estos sistemas, con lo que también puede tener cualquiera de los navegadores mas usados.
De esta manera podemos conseguir que prácticamente el 100% de los potenciales usuarios de la aplicación, no tengan ningún problema a la hora de acceder al sistema.
Conseguimos de esta forma, hacer una herramienta muy estandarizada, fácilmente accesible, competitiva, reutilizable.

Varios son los lenguajes que nos permiten esta finalidad anteriormente comentada.
Puesto que la aplicación tiene un peso importante en cuanto a su motor de funcionamiento, es necesario usar tecnologías que se ejecutan en la maquina remota (servidor) y posteriormente muestra los resultados en el cliente (host), la conocida tecnología cliente-servidor.
Encontramos tres tecnologías que desempeñan esta labor: PHP, ASP y Java (JSP, Servlets).
Nos decidimos sin lugar a duda por tecnología Java. Esta es libre y no presenta problemas de registro, cuando se implementa bajo el marco de la investigación y el desarrollo, siendo este el caso, en el que además no hay animo de lucro.
Pero no es este el aspecto más importante. Sin duda Java, es una tecnología con un respaldo a nivel mundial muy importante. Esta soportada por infinidad de sistemas,

y hay una gran comunidad de desarrollo detrás que sirve como ayuda y pilar fundamental a la hora de justificar su uso. Hoy en día, podemos encontrar muchísimas aplicaciones, módulos, versiones, programadas en Java. Esta tecnología, orientada a objetos, proporciona una gran modularidad, así como reutilización de código. Posee una potente API, como documentación y guía, así como una gran cantidad de Módulos ya implementados por otros desarrolladores, en versiones libres de registro, dispuestas para ser reutilizadas. En nuestra aplicación usamos tres módulos ya desarrollados anteriormente, y gracias a esta tecnología, son fácilmente acoplables y se agradece su funcionalidad.
Usando Java, nos sentimos, como un conjunto de desarrolladores mas, que implementan código y ofrecen su funcionalidad a la comunidad.

La manera en que Java nos puede ayudar, en nuestro propósito de estandarizar y difundir la aplicación vía Web, es con una tecnología de Sun Microsistem, propietario y creador de Java. Esta es llamada J2EE, el estándar de Java, dedicado a todo el desarrollo de entornos Web e inteligencia de negocio.
Para nuestro proyecto hemos usado el modelo MVC (Model View Controller), como modelo de proyecto. Mediante este, conseguimos tener tres bloques separados, que interactúan entre sí: El Modelo, es el encargado de llevar toda la carga de proceso, el motor de busqueda, funcionamiento. La Vista, se encarga de mostrar los datos al usuario final, estos pueden ser mostrados como un formulario a rellenar por el usuario, o como un formulario ya rellenado por la aplicación, también elementos estáticos que forman la apariencia de nuestra web. El controlador, es el encargado de mover el flujo de datos entre la vista, o interfaz grafica, y el modelo o unidad de negocio.
Para cada una de estos bloques, usamos distintos elementos del estándar J2EE.
Esto son:
Vista:
Paginas estáticas: HTML (paginas web), CSS (capas estilo)
Paginas dinámicas: JSP (Encargadas de mostrar informacion dinamica)

Controlador:
Servlets, clases Java encargadas de mover el flujo de datos entre la vista y el modelo.

Modelo:
Clases Java puras, encargadas de llevar todo el volumen de negocio. Encargadas de crear las estructuras intermedias que nos ayudaran a mover la información.

Para almacenar y mover los datos, según avanza la aplicación, hemos decidido usar estructuras internas como tablas hash. Estructuras de datos implementadas ya por la Api de Java, las cuales proporcionan búsquedas y accesos muy rápidos a los datos. No hemos usado bases de datos, para hacer al sistema más ligero, con menos requisitos a la hora de ser implementado y montado sobre un servidor de aplicaciones.
Estos datos, son insertados en la sesión propia de cada instancia de la aplicación que este ejecutándose. De esta manera los datos persisten a través del avance de la aplicación.

Esta aplicación puede funcionar en cualquier navegador web estándar, como pude ser Explorer, Netscape, Mozilla, Conqueror, FireFox. De esta forma será muy difícil no tener un sistema capaz de poder acceder a nuestra aplicación y usar sus servicios.

Requisitos Software

Nuestra aplicación al estar programada completamente en Java, y poseer un código abierto, es fácilmente reconfigurable. Sus módulos pueden ser reutilizados, o adaptados según las distintas necesidades. Simplemente necesitaremos un entorno de desarrollo de Java, con capacidades de compilar. Hemos usado el JSDK1.5 para compilar nuestro código, y hemos usado la API perteneciente a esta versión para usar los métodos que Java nos proporciona.

Una vez tengamos la aplicación creada, esta puede ser desplegada en cualquier tipo de entorno para servidor de aplicaciones. Nosotros preferimos montar la aplicación sobre Tomcat, de Apache, que esta incluido dentro del proyecto de Jackarta. Por su característica de software libre y su mas que suficiente funcionalidad.
Por supuesto, Tomcat, así como otros servidores de aplicaciones como pueda ser WebSphere (IBM), Oracle Application Server (Oracle), ISS (Microsoft), ... , pueden desplegar en sus contenedores la aplicación. Dependiendo de cada uno de estos productos, pueden ir montados sobre un sistema operativo u otro. Pero la aplicación en si, montada sobre Tomcat, puede dar sus servicios desde cualquier plataforma, funcionando de esta manera, la maquina como el servidor que procesa y da respuesta al cliente.
Por todo esto, seguimos apostando por Jakarta Tomcat, como la herramienta más conveniente.

Requisitos Hardware

Como hemos visto anteriormente, hoy en día no es difícil encontrar una maquina capaz de funcionar como un servidor de Aplicaciones.
Podemos hacer funcionar Tomcat, bajo un cluster de varias maquinas, con varias CPUs cada una, en un entorno de producción con bastante carga, y es soportado perfectamente. De la misma forma, Tomcat, puede funcionar perfectamente en una maquina Pentium II a 350 Mhz, con 256 Mb de memoria RAM, sobre un sistema operativo Linux en una distribución Debian sin ningún problema.
Claramente la carga de trabajo que soporta uno u otro sistema no es la misma, pero eso ya no es problema de la aplicación, ni el servidor, eso es mas un problema de la entidad que quiera ofrecer los servicios y el numero de usuarios que quiere soportar simultáneamente.
Vemos como se adapta perfectamente el producto a las distintas condiciones, y configuraciones, para las que en un futuro sea propuesto.

4 – Funcionamiento de la Aplicación

La aplicación se divide principalmente en dos partes diferenciadas, pero accesibles desde el mismo lugar.

Una de ellas nos permite la creación de metadatos.
Se selecciona el LO (disco físico local o en máquina remota a través de URL), y se elige un esquema de clasificación de metadatos. A continuación se muestra el formulario del esquema seleccionado, dónde el usuario introduce los valores correspondientes.

Estos datos se pueden empaquetar en un archivo en formato “.zip”, con el mismo nombre que tiene el LO en su situación original. Dentro del paquete se incluyen el LO (copia del archivo original) y los archivos en RDF y XML que, codificados según nuestros esquemas, contienen los metadatos asociados al LO. La nomenclatura del fichero de RDF comenzará por “LM_”, “LE_” o “DC_” según corresponda al esquema

LOM, LOMe o DC respectivamente, seguido del nombre que tiene el LO, con formato ".rdf". El fichero XML, tiene la misma nomenclatura, pero su extensión es ".xml".

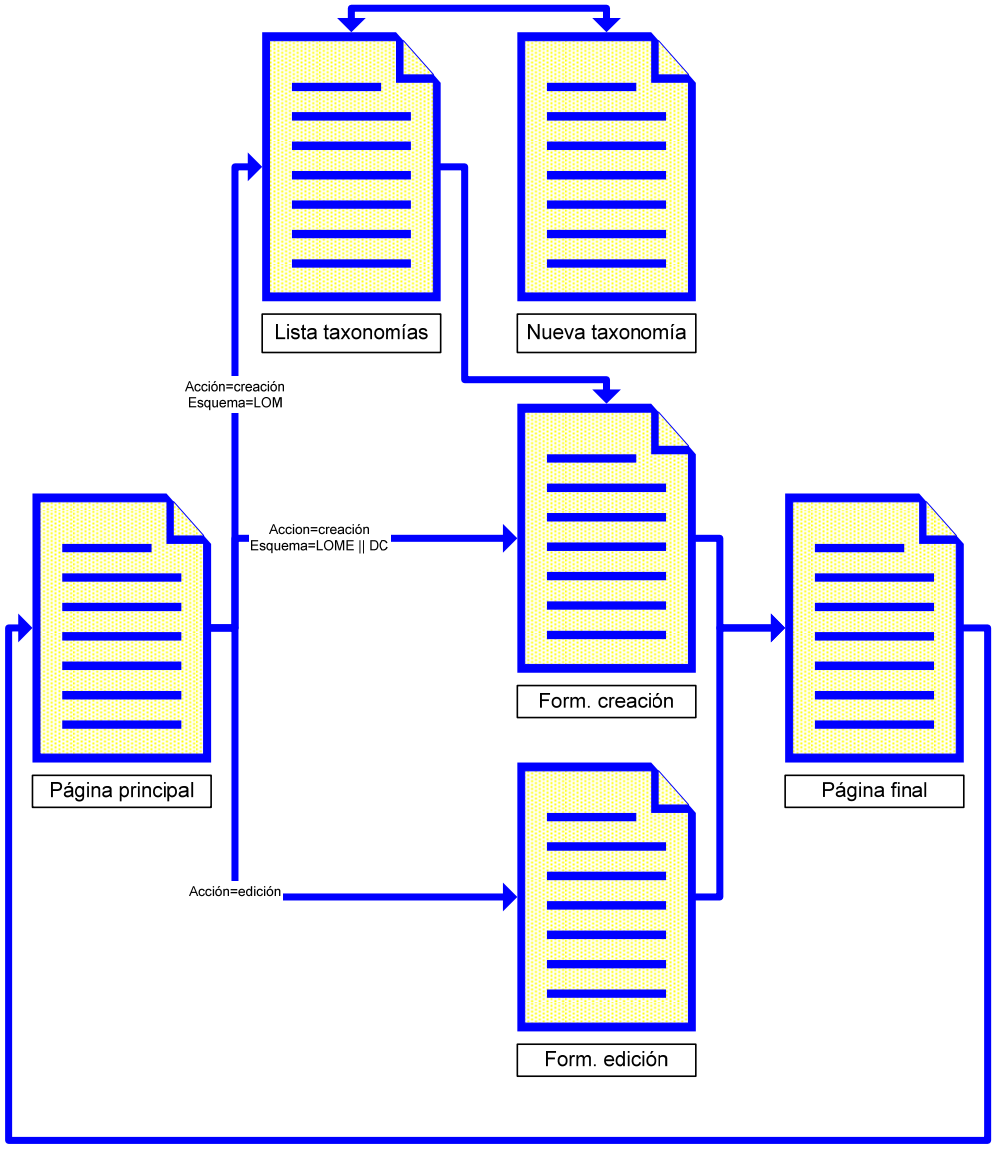
La otra parte, comprende la edición de metadatos.

En este apartado el usuario podrá visualizar y modificar los valores de los metadatos que se introdujeron al clasificar un LO en nuestra herramienta.

Se parte, por tanto, de un archivo comprimido ".zip" con el contenido ya comentado. Se selecciona el archivo comprimido y a continuación, se muestra el formulario con los valores que se rellenaron por última vez al clasificar el LO. Se puede, en este punto modificar o añadir valores, cumpliendo siempre las normas establecidas. Estos datos actualizados, se pueden guardar de la misma manera que en la creación; en un ".zip", que puede ser editado en siguientes ocasiones.

4.1 - Interfaz gráfica y uso

A continuación se analizan las diferentes páginas que soportan esta doble funcionalidad. Se analizan por separado los caminos de creación y de edición de metadatos que parten de la misma página (página principal) y que finalizan también en una común (página final).



4.1.1 - Página principal

Se muestra una breve descripción del proyecto, y los formularios para recoger los datos necesarios para la creación y edición de metadatos.

Si es la primera vez que interactuamos con la aplicación se entraría por la opción de creación para asociar metadatos a nuestro LO desde el principio, si queremos modificar o visualizar los valores asociados a un LO conocido elegiríamos la edición.

4.1.2 - Creación de Metadatos

En el formulario de creación se debe introducir el nombre del LO (fichero en cualquier formato), el origen del mismo (disco físico o URL, nuestra aplicación busca el LO donde le hayamos indicado) y el esquema de los metadatos que

queremos asociar. Una vez introducidos, se puede comenzar el proceso de creación, clasificando el LO de acuerdo al esquema seleccionado.
Dependiendo del esquema iremos directamente a los formularios de metadatos o bien habrá un análisis de taxonomías previo.

Esto último sucede cuando se selecciona LOM. En este caso, la aplicación nos lleva a un menú dónde podemos seleccionar una taxonomía controlada por nuestra aplicación (lista de taxonomías), crear una nueva (crear taxonomía) o importar una ya creada externa al proyecto. Una vez seleccionada la taxonomía, se accede al formulario de metadatos (formulario de creación de metadatos).

A esta misma pantalla, pero con metainformación acorde con el esquema de e-aula iremos directamente desde la principal al seleccionar esquema LOME, puesto que para este esquema existe una taxonomía creada de manera estática sobre el dominio de los lenguajes de programación. Lo mismo sucede si seleccionamos DC; pasamos directamente al formulario de metadatos de Dublin Core.

Posibles errores: Es posible que se produzcan errores a la hora de buscar la ruta especificada. Estos pueden ser producidos, por problemas en los permisos de lectura, o porque la dirección especificada no exista. Del mismo modo, debemos tener cuidado, no insertar una URL y especificar en los selectores del tipo, como archivo, o viceversa.

4.1.3 - Lista de taxonomías

Por un lado, se muestra el propósito de cada una de las taxonomías que posee la aplicación, el usuario puede seleccionar la que desee y directamente se accede al formulario de metadatos LOM con esta información cargada en la categoría classification.

Por otro lado, se da la opción de crear una taxonomía desde el principio. Si se selecciona esta opción se dirige a la página de creación de taxonomías.

4.1.4 - Creación de taxonomías

En esta página el usuario va a crear su clasificación desde el principio siguiendo el estándar marcado por LOM.

Se deben completar los campos de fuente, propósito y nombre de la taxonomía y de manera dinámica se puede crear el árbol taxonómico que representará su clasificación. A continuación se explica la estructura de la taxonomía a crear.

Una taxonomía siguiendo el estándar LOM se compone de taxones, cada uno de los cuales se compone a su vez de ID, ENTRY (un identificador y un label que lo describen) y otra entrada TAXON, ya que un taxon puede englobar a otros.

```
taxon
        id: I
        entry: Informática
        taxon
                id: S
                entry: Sistemas
                taxon
                        id: R
                        entry: Redes
                        taxon
                taxon
        taxon
taxon
```

Usando este estándar podemos obtener una clasificación jerárquica formada por diferentes niveles de taxones. La estructura descrita se acopla perfectamente a un árbol, esta es la forma visual que van tomando los datos introducidos.

Los taxones son los elementos fundamentales dentro de esta estructura. En nuestro árbol, un taxon puede o bien estar cerrado (no tiene asociado id, entry, taxon) o bien estar abierto, y tener atributos asignados. Inicialmente los taxones están cerrados, y una vez que se abren (mediante creación) no pueden volver al estado inicial.

Todas las acciones que el usuario puede ejecutar para crear el árbol, van dirigidas contra los taxones, se pueden crear, modificar y borrar. El taxon sobre el que se quiera actuar debe estar seleccionado (un clic de ratón).

Para “crear” un taxon debemos seleccionar un taxon cerrado y dar al botón “crear”. Aparecen dos campos de texto dónde podemos escribir el id y el entry del taxon seleccionado. Se da a “grabar” y en el árbol de la izquierda quedan añadidos los valores. En este momento el taxon queda abierto. Además de los atributos id y entry también se crean dos nuevos nodos taxon cerrados, uno como atributo, supondría un posible hijo del taxon seleccionado y otro a nivel del taxon seleccionado, posible hermano. De tal forma que se puede continuar con la jerarquía en todos los niveles.

La modificación debe realizarse sobre taxones abiertos, al igual que el borrado (por consistencia de la estructura no se dejan borrar nodos cerrados).

Para modificar los atributos de un taxon existente, le damos a modificar. Nos aparecen en los dos campos de texto los valores actuales del taxon seleccionado, los modificamos y le damos a cargar, automáticamente se modifican los valores.

Por último, se puede borrar un nodo taxon, le damos a borrar y se elimina el seleccionado y los taxones hijos con todos sus valores.

Se recomienda utilizar una letra (sigla) en el campo ID, ya que luego la concatenación de ids de las diferentes ramas del árbol (separados por puntos) formarán los valores finales de la clasificación. En la inserción y modificación, se controla que no se inserten dos ids iguales para taxones hermanos a cualquier nivel.

Una vez que el usuario estima que la taxonomía está completa se puede incorporar la misma a las manejadas por la aplicación pulsando el botón de “generar taxonomía”. En un proceso transparente al usuario, se ha creado un fichero usando nuestro estándar de RDF para taxonomías con la información presente en el árbol y se ha incluido la nueva taxonomía, en la lista de disponibles para el usuario.

De hecho se redirige a la página que lista las taxonomías creadas, dónde se puede comprobar esto último.

4.1.5 - Formulario creación metadatos (DC, LOM o LOMe)

Estos son los tipos de formulario que podemos encontrar para la inserción de metadatos. Se presentan, en el caso de DC y LOM todos los atributos que forman los esquemas. En DC de manera plana y en LOM divididos por categorías, para hacer más fácil su inserción. En el caso del esquema de e-aula también se encuentra dividido en categorías al estilo de LOM.

En ocasiones, es necesario que los campos se rellenen siguiendo un estándar específico. Los metadatos que pueden ofrecer lugar a dudas están acompañados de una ayuda explicativa que se despliega al pinchar en el símbolo de información que los acompaña. Dependiendo de la naturaleza de los mismos se presenta un select con los valores predefinidos o bien un campo de texto vacío, sólo es necesario rellenar los campos, que consideramos oportunos. El serializado tratará únicamente los valores de los campos rellenos.

Hay categorías, como lifecycle que controla el ciclo de vida del LO, que pueden tener más de un juego de metadatos completo. Estos diferentes juegos pueden ser añadidos en tiempo de edición. Las categorías que poseen este rasgo son: Lifecycle, Annotation y Relation, y se impone una restricción sobre los metadatos de las mismas; es necesario rellenarlos todos, o no rellenar ninguno. Se muestra un mensaje de error en caso contrario.

Una vez, tengamos todos los metadatos insertados. Simplemente tendremos que pulsar sobre el botón de enviar, y si todo el proceso ha sido correcto, tendremos en el directorio por defecto, adjuntado un LO, con su documento RDF especificado según el esquema seleccionado, y el archivo XML.

Se dirige el control a la página final.

Posibles errores: Nuestra aplicación realiza una comprobación de los valores insertados, controlando que siguen las normas de codificación establecidas. Así mismo hay ciertos campos, que dependen de otros, esta comprobación también será realizada y comunicada como error en caso de no cumplirse la correspondencia entre unos valores y otros. De esta manera, la aplicación es robusta con respecto a los posibles valores introducidos por el usuario.

En ningún punto la aplicación deja de funcionar, son errores completamente controlados y capturados. El programa simplemente los notifica, permitiéndonos volver a realizar el proceso de forma correcta.

4.1.6 - Edición de Metadatos

En el formulario de edición se debe introducir el nombre del “.zip” asociado al LO que queremos acceder y el origen del mismo. El esquema de metadatos se maneja internamente dependiendo del contenido fichero comprimido.

Le damos a “Aceptar” y nos dirige directamente al formulario (formulario edición de metadatos). El formulario tiene el mismo formato que el de creación, salvo alguna particularidad.

4.1.7 - Formulario edición de metadatos (DC, LOM o LOMe)

La idea y el formato es el mismo que en el formulario de creación, aunque en este caso se muestran los valores que el usuario adjuntó al LO en su creación, dependiendo del esquema que seleccionó.

Algunos aspectos a señalar:

- Para los casos de Lifecycle, Annotation y Relation, si el usuario rellenó los valores, se muestra un nuevo juego de valores en blanco para que se inserte una nueva instancia de estas categorías y así realizar un mantenimiento del LO. Las normas de inserción son iguales al proceso de creación.
- En la categoría classification se muestra un select con los valores de la taxonomía que se seleccionó en el momento de la creación. Se puede seleccionar otro valor de esta clasificación, pero no elegir otra taxonomía.
- Se puede, por tanto, modificar los valores y finalmente volver a salvarlos, pulsando sobre el botón UPDATE. A esta altura, se conectará de nuevo con la lógica ya comentada, y los archivos de metadatos asociados al LO serán actualizados con las modificaciones.

Se dirige el control a la página final.

4.1.8 - Página final

Una vez validado todo el proceso anterior, sin ningún error. Se nos mostrará una página con cuatro opciones.

- Mostrar RDF: Se muestra en una nueva página web el archivo RDF serializado a partir de los datos introducidos.

- Mostrar XML: Se muestra en una nueva página web el archivo XML serializado a partir de los datos introducidos.

- Guardar en ZIP: Esta función nos permite guardar los ficheros generados en un archivo “.zip”. Se almacenan en el mismo, como ya hemos comentado anteriormente los ficheros de metadatos serializados en XML y RDF (los mismos que en las opciones de mostrado) y el LO.

- Volver sin hacer nada: Redirecciona a la página principal, para dar por finalizado el proceso de clasificación.

5 - Conclusión

La Enseñanza Asistida por Computador o e-learning, es un modo de enseñanza con muchas posibilidades y grandes promesas de futuro. Internet ha supuesto un empujón muy importante, pero la inyección que realmente necesita para que todo su potencial sea aprovechado es la personalización. La gestión de Objetos Educativos es la base, y aún una tarea pendiente. Creemos que la combinación de RDF y LOMe soluciona gran parte de las carencias que se encontraban a la hora de documentar OE para almacenarlos en repositorios.

MetaRDF es una aplicación generada bajo el gran peso que suponen los estándares en la vida moderna.

Es una herramienta que funciona via web, adaptada a cualquier tipo de navegador, accesible desde cualquier punto de acceso que tenga conexión con Internet. De esta forma nos aseguramos de tener una herramienta a disposición de cualquier persona con un punto de acceso.

La aplicación usa Java como lenguaje de programación. Este aspecto dota a la misma de un carácter altamente portable y reutilizable al ser un lenguaje de programación orientado a objetos. A su vez el modelo vista controlador que ha sido el patrón seguido para estructurar las clases de la herramienta, nos permite tener una alta modularidad la cual facilita la adaptación y reutilización del código para futuras versiones o mejoras.

Es importante el destacar el aspecto libre en cuanto a necesidades de compra de licencias de software o instalar bajo un sistema operativo especifico. Esto no ocurre pues la aplicación es multiplataforma y se ha programado bajo

La potencia de esta herramienta reside en cuatro puntos fundamentales:

- Herramienta Web, con acceso desde Internet.
- Programada en Java, con su tecnología orientada a cliente-servidor J2EE, accesible desde cualquier plataforma y navegador.
- Se codifican los metadatos con lenguajes de marcado(RDF y XML)
- Usamos los estándares LOM y DC mas el LOMe desarrollado por nosotros basándonos en LOM para la organización de los metadatos.

6 - Bibliografía

- RDF, metadatos, XML y la web semántica. Manuales de teoría en forma de diapositivas.
  http://www.w3.org/2000/Talks/1206-xml2k-tbl
- W3C y por que de RDF y OWL
  http://builderau.com.au/architect/work/0,39024596,20283034,00.htm
- Programa solo para hacer gráficos a partir de RDF
  http://www.w3.org/2001/11/IsaViz/
  http://www.dfki.uni-kl.de/frodo/RDFSViz/
- Primer de RDF al español
  http://www.sidar.org/recur/desdi/traduc/es/rdf/rdfsch.htm
- Sintaxis RDF en español
  http://www.sidar.org/recur/desdi/traduc/es/rdf/rdfesp.htm
- Tutorial RDF
  http://www.w3schools.com/rdf/default.asp
- Ejemplos RDF
  http://www.w3.org/2000/10/rdf-tests
- ¿Qué es RDF?
  http://www.xml.com/pub/a/2001/01/24/rdf.html
- Tutorial de RDF
  http://www.xulplanet.com/tutorials/mozsdk/rdfstart.php
- ¿Qué es un learning Object?
  http://www.masterdisseny.com/master-net/elearning/0015.php3
- Glosario de términos
  http://www.campusformacion.com/glosario.asp
- Artículos de Rob Koper
  http://www.unfold-project.net:8085/UNFOLD/general_resources_folder/papers/
- DCdot
  http://www.ukoln.ac.uk/metadata/dcdot/
- Reggie - The Metadata Editor
  http://metadata.net/dstc/
- MetaBrowser System
  http://metabrowser.spirit.net.au/
- Nordic DC metadata creator (including URN generator)
  http://www.lub.lu.se/cgi-bin/nmdc.pl
- http://www.imsproject.org
- http://dublincore.org/ltsc.ieee.org/wg12/files/
  LOM_1484_12_1_v1_Final_Draft.pdf
- http://www.mindspring.com/~wason/TDW/drtomtaxonomiesguidecas.html

Los desarrolladores de este proyecto, Patricia Ferrer Medina, Pablo Gámez Guillén y Santiago Alonso Domínguez, autorizamos a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales, tanto la propia memoria como el código fuente, la documentación y el prototipo desarrollado.

Madrid, a 1 de Julio de 2005

Patricia Ferrer Medina

Pablo Gámez Guillén

Santiago Alonso Domínguez